E.U.I.T.T.

PROYECTO FIN DE CARRERA

Ecualización y Optimización de la Codificación para la Simulación de Sistemas OFDM

Autor: Daniel Gallego Blanco

Tutor: José Enrique González García

Septiembre 2013

E.U.I.T. TELECOMUNICACIÓN

PROYECTO FIN DE CARRERA
PLAN 2000

TEMA: Simulación y procesado en comunicaciones

TÍTULO: Ecualización y optimización de la codificación para la simulación de sistemas OFDM

AUTOR: Daniel Gallego Blanco

TUTOR: José Enrique González García **Vº Bº.**

DEPARTAMENTO: DIAC

Miembros del Tribunal Calificador:

PRESIDENTE: Gregorio Rubio Cifuentes

VOCAL: José Enrique González García

VOCAL SECRETARIO: Carlos Rueda Frías

DIRECTOR:

Fecha de lectura: 30 de septiembre de 2013

Calificación: **El Secretario,**

RESUMEN DEL PROYECTO:

En este proyecto de ingeniería se desarrollan los conceptos teóricos necesarios para poder comprender tanto la modulación OFDM, como los esquemas que la acompañan en un sistema completo:codificación, ecualización, modulación, etc. Todo con la finalidad de poder simular un esquema completo OFDM. La simulación se va a realizar gracias a las herramientas que nos proporciona Matlab.

Mediante el programa y la interfaz gráfica desarrollada podemos realizar diversas simulaciones con las que comprender nuestro sistema. Los objetivos fundamentales dentro de la simulación del sistema es poner a prueba el empleo de turbo códigos (comparándolo con los códigos convolucionales tradicionales) y de un ecualizador. Todo ello con la intención de optimizar nuestro sistema OFDM en condiciones cada vez más adversas.

Las simulaciones realizadas en el proyecto nos aportan aproximaciones a los resultados que se obtendrían a un sistema real y conclusiones sobre su funcionamiento respecto a las condiciones de canal a las que se somete.

RESUMEN

Los sistemas basados en la técnica OFDM (Multiplexación por División de Frecuencias Ortogonales) son una evolución de los tradicionales sistemas FDM (Multiplexación por División de Frecuencia), gracias a la cual se consigue un mejor aprovechamiento del ancho de banda. En la actualidad los sistemas OFDM y sus variantes ocupan un lugar muy importante en las comunicaciones, estando implementados en diversos estándares como pueden ser: DVB-T (estándar de la TDT), ADSL, LTE, WIMAX, DAB (radio digital), etc. Debido a ello, en este proyecto se implementa un sistema OFDM en el que poder realizar diversas simulaciones para entender mejor su funcionamiento. Para ello nos vamos a valer de la herramienta Matlab.

Los objetivos fundamentales dentro de la simulación del sistema es poner a prueba el empleo de turbo códigos (comparándolo con los códigos convolucionales tradicionales) y de un ecualizador. Todo ello con la intención de mejorar la calidad de nuestro sistema (recibir menos bits erróneos) en condiciones cada vez más adversas: relaciones señal a ruido bajas y multitrayectos. Para ello se han implementado las funciones necesarias en Matlab, así como una interfaz gráfica para que sea más sencillo de utilizar el programa y más didáctico.

En los capítulos segundo y tercero de este proyecto se efectúa un estudio de las bases de los sistemas OFDM. En el segundo nos centramos más en un estudio teórico puro para después pasar en el tercero a centrarnos únicamente en la teoría de los bloques implementados en el sistema OFDM que se desarrolla en este proyecto. En el capítulo cuarto se explican las distintas opciones que se pueden llevar a cabo mediante la interfaz implementada, a la vez que se elabora un manual para el correcto uso de la misma. El quinto capítulo se divide en dos partes, en la primera se muestran las representaciones que puede realizar el programa, y en la segunda únicamente se realizan simulaciones para comprobar que tal responde nuestra sistema a distintas configuraciones de canal, y las a distintas configuraciones que hagamos nosotros de nuestro sistema (utilicemos una codificación u otra, utilicemos el ecualizador o el prefijo cíclico, etc…). Para finalizar, en el último capítulo se exponen las conclusiones obtenidas en este proyecto, así como posibles líneas de trabajo que seguir en próximas versiones del mismo.

ABSTRACT

Systems based on OFDM (Orthogonal Frequency Division Multiplexing) technique are an evolution of traditional FDM (Frequency Division Multiplexing). Due to the use of OFDM systems are achieved by more efficient use of bandwidth. Nowadays, OFDM systems and variants of OFDM systems occupy a very important place in the world of communications, being implemented in standards such as DVB-T, ADSL, LTE, WiMAX, DAB (digital radio) and another more. For all these reasons, this project implements a OFDM system for performing various simulations for better understanding of OFDM system operation. The system has been simulated using Matlab.

With system simulation we search to get two key objectives: to test the use of turbo codes (compared to traditional convolutional codes) and an equalizer. We do so with the intention of improving the quality of our system (receive fewer rates of bit error) in increasingly adverse conditions: lower signal-to-noise and multipath. For these reasons necessaries Matlab´s functions have been developed, and a GUI (User Graphical Interface) has been integrated so the program can be used in a easier and more didactic way.

This project is divided into five chapters. In the second and third chapter of this project are developed the basis of OFDM systems. Being developed in the second one a pure theoretical study, while focusing only on block theory implemented in the OFDM system in the third one. The fourth chapter describes the options that can be carried out by the interface implemented. Furthermore the chapter is developed for the correct use of the interface. The fifth chapter is divided into two parts, the first part shows to us the representations that the program can perform, and the second one just makes simulations to check that our system responds to differents channel configurations (use of convolutional codes or turbo codes, the use of equalizer or cyclic prefix…). Finally, the last chapter presents the conclusions of this project and possible lines of work to follow in future versions.

Agradecimientos

A mi familia (en especial
a David y Óscar) y a
todos aquellos que
han estado siempre
a mi lado apoyándome
cuando lo he necesitado.

ÍNDICE DE CONTENIDOS

CAPÍTULO 1

INTRODUCCIÓN

Los sistemas OFDM (Multiplexación por División de Frecuencias Ortogonales) surgen para solucionar el problema que existía con los tradicionales métodos de FDM (Multiplexación por División de Frecuencia), donde había que dejar una región de espectro libre entre ellas para que funcionaran correctamente. Desde un punto de vista de aprovechamiento de ancho de banda no resultaba eficiente. Es por ello que se proponen los sistemas OFDM, donde las portadoras están separadas entre sí un margen que hace que sean ortogonales entre ellas, dejando de resultar necesario dejar un margen de espectro libre, subiendo por lo tanto la eficiencia de aprovechamiento espectral (el espectro del que disponemos es un bien limitado). En la actualidad los sistemas OFDM y sus variantes ocupan un lugar muy importante en las comunicaciones, estando implementados en diversos estándares como pueden ser: DVB-T (estándar de la TDT), ADSL, LTE, WIMAX, DAB (radio digital), etc. Debido a ello, en este proyecto se implementa un sistema OFDM en el que poder realizar diversas simulaciones para entender mejor su funcionamiento. Para ello nos vamos a valer de la herramienta Matlab.

Los objetivos fundamentales dentro de la simulación del sistema es poner a prueba el empleo de turbo códigos (comparándolo con los códigos convolucionales tradicionales) y de un ecualizador. Todo ello con la intención de mejorar la calidad de nuestro sistema (recibir menos bits erróneos) en condiciones cada vez más adversas: relaciones señal a ruido bajas y multitrayectos. Para ello se han implementado las funciones necesarias en Matlab, así como una interfaz gráfica para que sea más sencillo de utilizar el programa y más didáctico.

En los capítulos segundo y tercero de este proyecto se efectua un estudio de las bases de los sistemas OFDM. En el segundo nos centramos más en un estudio teórico puro para después pasar en el tercero a centrarnos únicamente en la teoría de los bloques implementados en el sistema OFDM que se desarrolla en este proyecto. En el capítulo cuarto se explican las distintas opciones que se pueden llevar a cabo mediante la interfaz implementada, a la vez que se elabora un manual para el correcto uso de la misma. El quinto capítulo se divide en dos partes, en la primera se muestran las representaciones que puede realizar el programa, y en la segunda únicamente se realizan simulaciones para comprobar que tal responde nuestra sistema a distintas configuraciones de canal, y las a distintas configuraciones que hagamos nosotros de nuestro sistema (utilicemos una codificación u otra, utilicemos el ecualizador o el prefijo cíclico, etc…). Para finalizar, en el último capítulo se exponen las conclusiones obtenidas en este proyecto, así como posibles líneas de trabajo que seguir en próximas versiones del mismo.

CAPÍTULO 2

ESTUDIO TEÓRICO

En este capítulo se van analizar los conceptos teóricos básicos utilizados en el desarrollo de este proyecto, con la intención de ayudar a comprender el esquema desarrollado de simulación.

2.1 OFDM

2.1.1. Fundamentos

La técnica de transmisión OFDM no es más que un mecanismo de transmisión multiportadora en el cual se multiplexan un conjunto de símbolos sobre un conjunto de subportadoras. Debido a que estas subportadoras son ortogonales, se pueden transmitir simultáneamente todos los símbolos manteniendo la capacidad de separación de los mismos en el receptor. Ya que se han definido las subportadoras como ortogonales, aprovechamos para recordar la definición de ortogonalidad: se dice que dos subportadoras $\phi_i(t)$ y $\phi_k(t)$ son ortogonales si se cumple dado el intervalo Ts:

(2.1)

$$\int_0^{Ts} \phi_i(t)\phi_k^*(t)\,dt = \begin{cases} 1 \;\; si \;\; i = k \\ - \\ 0 \;\; si \;\; i \neq k \end{cases}$$

La señal temporal de cada subportadora puede formularse en banda base como:

(2.2)

$$x_k(t) = \; e^{j2\pi k\Delta f t} \cdot rect_{T_s}$$

Siendo Δf la distancia entre las subportadoras (por lo que la frecuencia de cada subportadora es $k\Delta ft$) y $rect_{T_s}$ un pulso rectangular de duración T_s

Si tenemos un conjunto de k símbolos complejos $(d_0, d_1, d_2 \ldots d_k)$ que deseamos modular mediante las subportadoras de (2.2), tendremos el símbolo OFDM:

(2.3)

$$s(t) = \; \sum_{k=0}^{K-1} d_k e^{j2\pi k\Delta f t} \cdot rect_{T_s}$$

Esta señal sería en banda base pero en un sistema real se modularía para subir a la frecuencia f_0 obteniendo:

(2.4)

$$s(t) = \; \sum_{k=0}^{K-1} d_k e^{j2\pi k(f_0+\Delta f t)} \cdot rect_{T_s}$$

Comprobamos que se cumple la condición de ortogonalidad (2.1) en (2.2) para demostrar que efectivamente las subportadoras son ortogonales:

$$\int_0^{Ts} x_m(t)x_n^*(t)\,dt = \; \int_0^{Ts} e^{j2\pi\Delta f t(m-n)}dt = \; \int_0^{Ts} e^{j2\pi(m-n)t/T_s}dt =$$

$$= \frac{\sin\big(2\pi(m-n)\big) + \cos\big(2\pi(m-n)\big) - 1}{2\pi(m-n)} = \begin{cases} 1 \;\; si \;\; m = n \\ - \\ 0 \;\; si \;\; m \neq n \end{cases}$$

Efectivamente se cumple la condición de ortogonalidad. Esto es debido al empleo de una frecuencia de separación que es la inversa al tiempo de símbolo ($\Delta f = 1/T_s$)

2.1.2. OFDMA

La técnica de acceso múltiple OFDMA (*Ortogonal Frequency Division Multiple Access*) utilizada, por ejemplo, en el estándar LTE, surge de forma natural a partir de la modulación OFDM al considerar la opción de distribuir los símbolos de información de cada usuario en distintas subportadoras. De esta forma, en un único símbolo OFDM se pueden transmitir simultáneamente la información de varios usuarios. En la figura 2.1 se representan *U* flujos de información correspondientes a los usuarios, siendo *Nk* el número de símbolos enviado para el usuario *k*-ésimo. En recepción se deberán separar el contenido de las subportadoras de cada usuario de las demás para poder recuperar la información original.

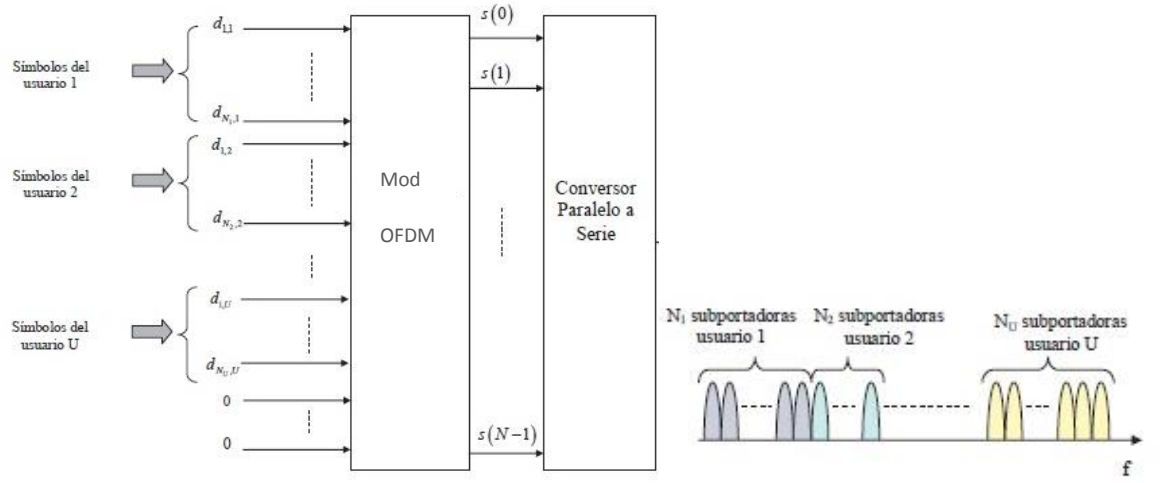

Figura 2.1.Ejemplo OFDMA

2.1.3. Ventajas

La modulación OFDM presenta múltiples ventajas:

- Elevada eficiencia espectral al utilizar múltiples portadoras que pueden ser ubicadas en un ancho de banda muy reducido.
- Robustez frente a las diferencias de retardo (a diferencia de los sistemas de una única portadora). La diferencia del retardo en el canal provoca interferencias entre símbolos que, a su vez, limitan la velocidad de los datos.
- Facilidad de ecualización en el dominio de la frecuencia. Esta ecualización es más sencilla de implementar que la tradicional en el tiempo.
- Uso de portadoras piloto. Son portadoras cuya información conoce el receptor, por lo que se puede analizar la respuesta del canal y corregir los efectos del medio.
- En un canal que no tenga una gran variación temporal, se puede realizar una modulación adaptativa según la SNR (relación señal a ruido) de cada subportadora.
- Menor sensibilidad a la sincronización temporal que las modulaciones monoportadora.
- Posibilidad de variar las velocidades superior e inferior, según el empleo de más o menos

portadoras en función del propósito de la comunicación. Esto puede resultar útil, por ejemplo, en comunicaciones móviles para dar una distinta *QoS* (calidad de servicio) a sus clientes.

- Implementación simple y eficiente gracias a la utilización de los algoritmos de IFFT y FFT para la modulación y demodulación respectivamente[1].

2.1.4. Desventajas

- PAPR (*Peak Average Power Ratio*) alta. Es decir, una gran diferencia entre la potencia máxima y la potencia media de la señal modulada, lo que dificulta la elección de un punto de trabajo en potencia que puede provocar distorsión no lineal en el transmisor. Esta distorsión puede aparecer tanto dentro como fuera de banda.
- Es muy sensible a los errores de frecuencia, que puede destruir la ortogonalidad. Esto es la desventaja más importante de esta modulación (aunque se puede combatir). En la siguiente figura se muestra la perdida de ortogonalidad entre subportadoras por desvío de fase:

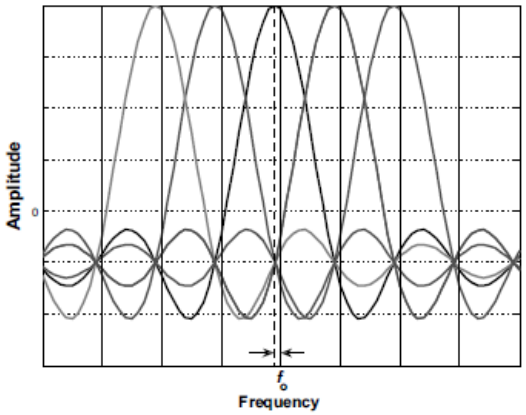

Figura 2.2.Ejemplo pérdida de ortogonalidad.

- La necesidad de los intervalos de guarda con prefijo cíclico[2] suponen en coste en energía que acaba suponiendo una menor eficiencia en energía.

2.2 Codificación de línea

Con la codificación se busca añadir redundancia a las señales para corregir errores que se hayan podido ocasionar en el trayecto. Eso sí, a costa de aumentar el número de bits a transmitir. Se va a proceder al estudio de los dos códigos utilizados en este proyecto: códigos convolucionales y turbo códigos.

2.2.1 Códigos convolucionales

Lo primero que hay que decir sobre la codificación convolucional es que estamos hablando de una codificación que opera tanto sobre la información actual, presente a la entrada del codificador,

[1] Esto se demostrará en el tema 3.
[2] El prefijo cíclico será explicado en el apartado 2.4.

como sobre la información pasada, por lo que se trata de un proceso con memoria (a diferencia de los sistemáticos).

Se definen tres parámetros fundamentales (*n, k, m*) para especificar un código convolucional, donde *n* es el número de bits a la salida del codificador, *k* el número de bits a la entrada de éste y *m,* el número de registros de memoria. Si dividimos *k* entre *n* obtenemos la relación o tasa de código que proporciona una medida de la eficiencia de codificación. A estos tres parámetros les podemos añadir un cuarto *L*, llamado longitud de constricción y que se define como:

(2.5)

$$L = k(m-1)$$

L representa el número de bits en la memoria del codificador, previos al bit actual de entrada, que afectan a la generación del código.

En la figura 2.3 podemos ver un ejemplo de un codificador convolucional [Benedetto 87]:

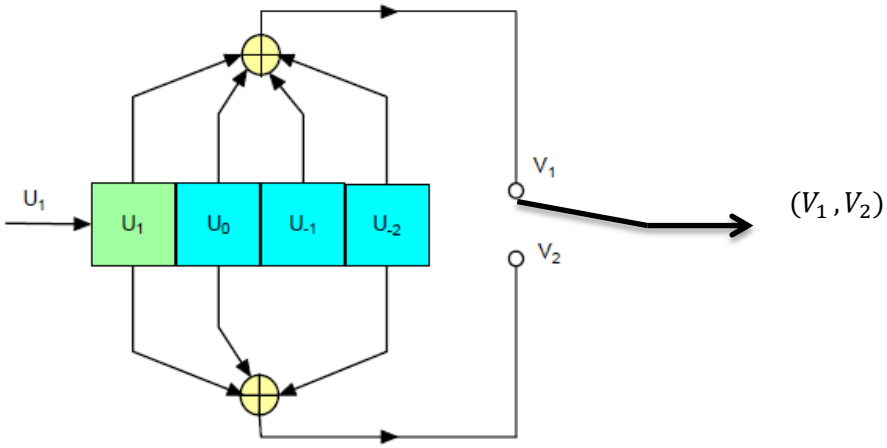

Figura 2.3. Estructura de un codificador convolucional simple

Este codificador está constituido por cuatro registros que almacenan los bits y dos sumadores en los que se combinan los bits de los registros. El código de salida se obtiene conmutando secuencialmente las dos salidas de los sumadores durante el periodo de cada bit de entrada. El primer registro (en color verde) contiene al bit actual U_1, mientras que los otros tres contienen a los bits anteriores U_0, U_{-1} y U_{-2}. Los bits de salida para este codificador son:

$$V_1 = U_1 + U_0 + U_{-1} + U_{-2}$$

$$V_2 = U_1 + U_0 + U_{-2}$$

Este codificador genera un código con *n*=2, *k*=1, *m*=4 y L=3. Por lo tanto, la tasa de código es de 1/2, lo que quiere decir que por cada bit de entrada se generan dos de salida.

Vamos a aprovechar el ejemplo de la figura 2.3 para proceder a explicar los tres métodos de definir una secuencia convolucional: diagrama de estados, diagrama de árbol y diagrama de trellis[3].

1) Diagrama de estados

   En el ejemplo anterior la longitud de constricción es tres, lo que quiere decir que hay tres bits anteriores al actual (U_1) en los registros de desplazamiento ($U_0 + U_{-1} + U_{-2}$). Debido a ello hay ocho posibles combinaciones diferentes de estos tres bits que determinan la secuencia de salida (V_1, V_2). La cantidad de combinaciones que podemos formar con los bits previos al actual se denomina como *estados de código* y están dados por la siguiente relación:

$$Número\ de\ estados = 2^L$$

2) Diagrama de árbol

   Si partimos de un estado inicial del registro de desplazamiento (0, 0, 0, 0), la salida es (0,0) y es posible obtener el código de salida para cada nuevo bit de entrada. Por ejemplo si el bit de entrada es cero, se ve que la salida sigue manteniéndose en (0,0). Si el bit de entrada es un 1, el estado del registro será (1, 0, 0, 0) y la salida será ahora (1, 1). Si el siguiente bit de entrada es un 0, el contenido del registro es (0, 1, 0, 0) y la salida (1, 1). Si por el contrario, el bit de entrada es un 1, el registro contendrá ahora (1, 1, 0, 0) y la salida será (0,0). El análisis continuado en esta forma da lugar a un diagrama de árbol en que el número de ramas aumenta cada vez que entra un nuevo bit al registro. Para el caso del codificador de la figura 2.3, el diagrama de árbol tiene la forma mostrada en la figura 2.4.

[3] El término trellis significa celosía, similar a la utilizada para soportar plantas trepadoras.

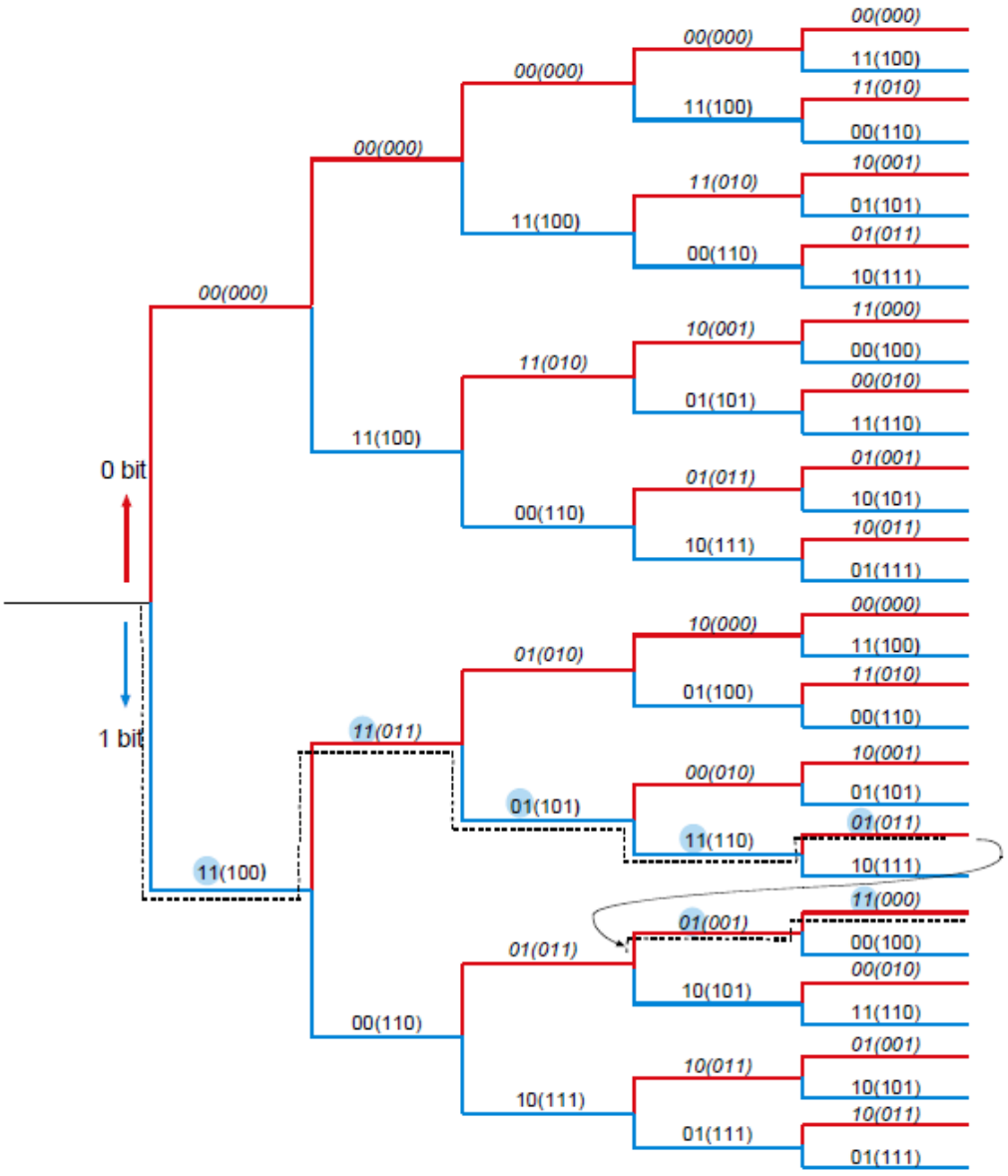

Figura 2.4. Diagrama de árbol para el decodificador de la figura 2.3

3) Diagrama de trellis

Con el diagrama de árbol el número de ramas crece como función de 2^L, pero si se observa la figura 2.4, se aprecia que la estructura se repite a partir de la tercera ramificación, es decir, se repite cada *L* ramificaciones. Sirviéndonos de esta propiedad se puede realizar una representación más adecuada. Esta representación se denomina diagrama de trellis. Este diagrama suele ser el preferido en la representación gráfica de la codificación convolucional. En la dirección horizontal se representan los instantes de tiempo discreto o de reloj, mientras que en el eje vertical se representan todos los bits del registro de desplazamiento sin incluir el actual. Si partimos de un estado determinado, hay dos posibles salidas dependiendo de si el bit de entrada es un 1 o un 0. En el diagrama estos posibles estados se indican mediante líneas continuas (si el bit de entrada es 0) o punteadas (si es un 1).

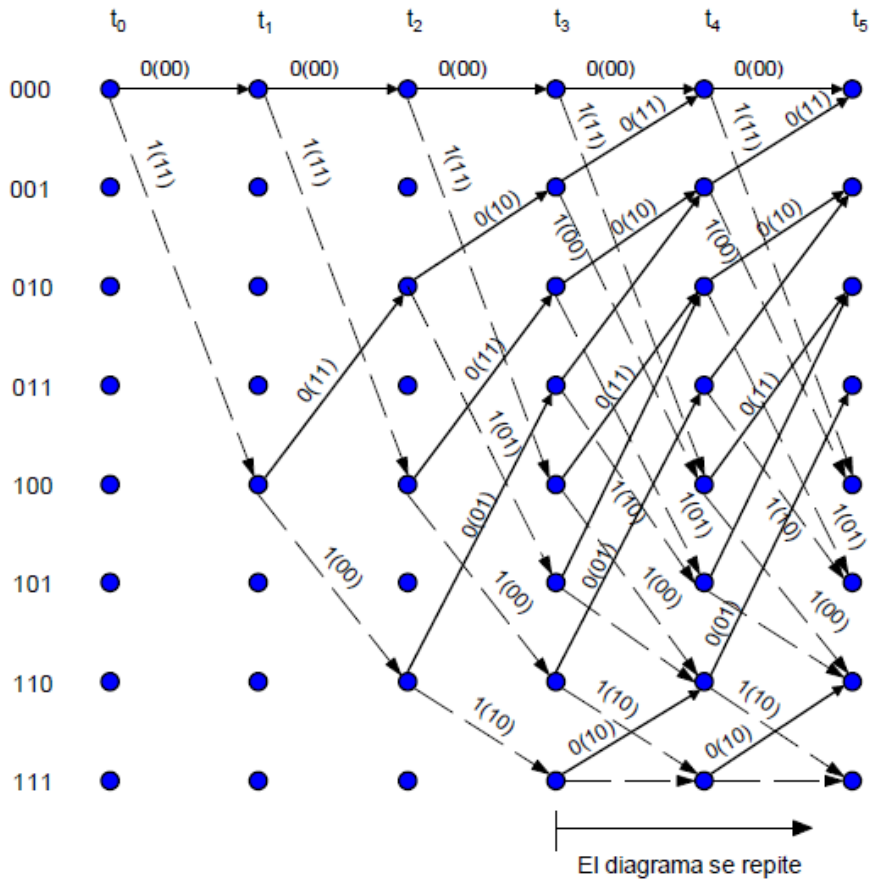

Figura 2.5. Diagrama de trellis para el codificador de la figura 2.3

Fijándonos en este diagrama se pueden obtener algunas propiedades de importancia para la corrección de errores en la decodificación. Así, si por ejemplo, en el instante *t0* el código de salida fue 00, esto implica que en el instante *t1* el código de salida será 00, si el bit a la entrada es un 0, o bien 11 si el bit a la entrada es un 1, pero no podrá ser 01 ni 10. Es decir, de un estado 00, el codificador sólo puede pasar al estado 00 o 01. Si en el proceso de decodificación después de un estado 00 se tiene un estado 01 o 10, será indicación de que ha habido un error en la transmisión y será necesario corregirlo, siendo tratado el tema de la corrección por el decodificador.

Para la decodificación se utiliza el algoritmo de Viterbi [Viterbi 79]. Este algoritmo lo que realizada es decodificación designada como de *máxima similitud*. La base de la decodificación de Viterbi radica en que, si dos trayectorias cualesquiera en el diagrama trellis dan lugar a un mismo estado de salida, siempre puede eliminarse una de ellas en la búsqueda de la trayectoria óptima. La trayectoria que se mantiene se designa como trayectoria superviviente y equivale a elegir el símbolo con la métrica de máxima similitud o la mínima distancia métrica. Tenemos dos opciones de utilizar de métrica junto con el algoritmo de Viterbi: la distancia de Hamming o la Euclidea. En el primer caso (métrica de la distancia Hamming) estaremos trabajando con una decodificación *Hard* (detección a posteriori); mientras que si trabajamos con una métrica Euclidea se tratará de una decodificación *Soft* (detección a priori).

2.2.2 Turbo códigos

Los turbo códigos son una evolución de los códigos convolucionales, de hecho un turbo código está formado por dos o más codificadores RSC[4] y un cierto número de permutadores o *interleavers*. Según como combinemos estos codificadores obtendremos los distintos tipos de turbo códigos, los principales son los siguientes:

1) Concatenación Paralela de Códigos Convolucionales (PCCC)
   Esta es la arquitectura más importante, ya que es la más utilizada en multitud de estándares como, por ejemplo, UMTS, CDMA2000 o LTE. Debido a ello se explicará con algo más de detenimiento. El esquema general es el siguiente [Barbulescu 04]:

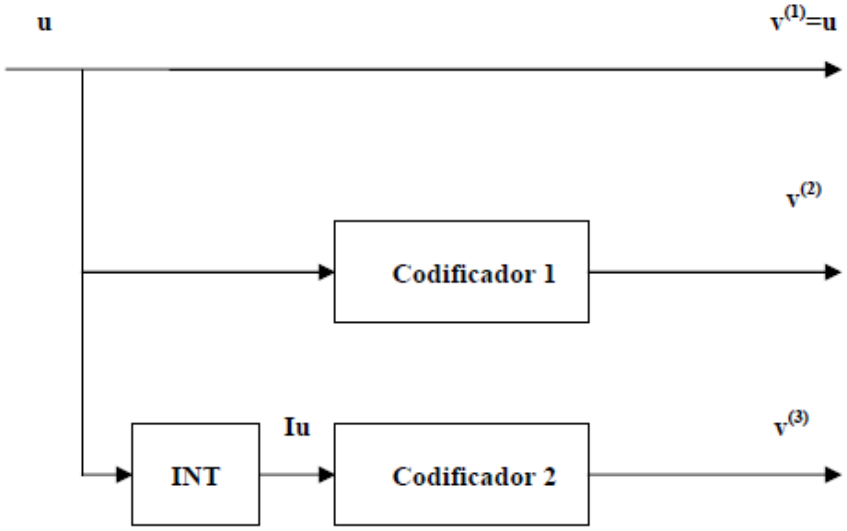

Figura 2.6. Arquitectura PCCC básica

En la figura 2.6 se observa que el turbo codificador está formado por dos codificadores convolucionales, un *Interleaver* (INT) y que una de las salidas está directamente unida a la entrada, lo que lo hace un codificador sistemático. El *Interleaver* toma la secuencia de información **u** y la reordena de manera pseudo-aleatoria y repetible, por lo que la secuencia **Iu** contiene exactamente la misma información pero en un orden totalmente distinto. El Codificador 1 toma la secuencia de información **u** y produce la de secuencia de paridad $\mathbf{v}^{(2)}$, mientras que el Codificador 2 hacer lo propio con **Iu**, produciendo la secuencia de paridad $\mathbf{v}^{(3)}$. Estas dos secuencias junto con la secuencia de información $\mathbf{v}^{(1)} = \mathbf{u}$, forman la salida del Turbo codificador, que está dada por [Ingvarsson 98]:

$$\mathbf{v} = (\mathbf{v}_1^{(1)}, \mathbf{v}_1^{(2)}, \mathbf{v}_1^{(3)}, \mathbf{v}_2^{(1)}, \mathbf{v}_2^{(2)}, \mathbf{v}_2^{(3)}, \dots, \mathbf{v}_k^{(1)}, \mathbf{v}_k^{(2)}, \mathbf{v}_k^{(3)})$$

Se observa que se producen tres bits de salida por cada uno de entrada, por lo que este turbo código tiene una tasa de 1/3. Los codificadores que forman parte del Turbo codificador no es necesario que sean iguales, aunque es lo habitual. En la siguiente figura

[4] Un codificador RSC es un codificador Convolucional Sistemático Recursivo (Recursive Systematic Convolutional)

se va a mostrar un codificador PCCC genérico, formado *n* codificadores convolucionales, *n*-1 *Interleavers* y una tasa de *n*+1 [Barbulescu 04].

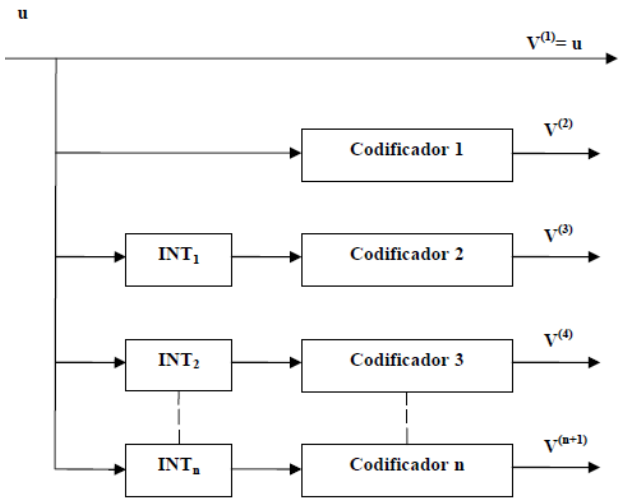

Figura 2.7. Arquitectura PCCC genérica

2) Concatenación Serial de Códigos Convolucionales (SCCC)

Una alternativa a la arquitectura PCCC es la estructura SCCC. En la figura 2.8 nos muestra un ejemplo de arquitectura SCCC cuya tasa de codificación es 1/3[Pietrobon 98].

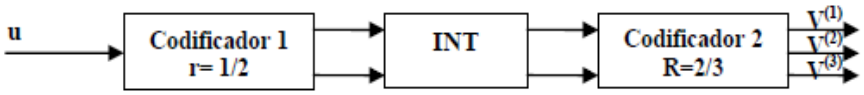

Figura 2.8. Arquitectura SCCC

El Codificador 1 se conoce como codificador externo y el Codificador 2 como codificador interno. Es importante resaltar la presencia del entrelazador entre los dos codificadores convolucionales, que porvoca efectos distintos en el desempeño de las arquitecturas PCCC y SCCC. Los esquemas PCCC presentan un mejor funcionamiento que los SCCC a bajas SNR (relación señal a ruido); sin embargo, con relaciones señal a ruido altas, los esquemas SCCC superan en prestaciones a los PCCC.

3) Concatenación Híbrida de Códigos Convolucionales (HCCC)

Otra alternativa es la estructura híbrida, llamada así debido a que es una combinación de la concatenación paralela y serial, como puede observarse en la figura 2.9[Pietrobon 98].

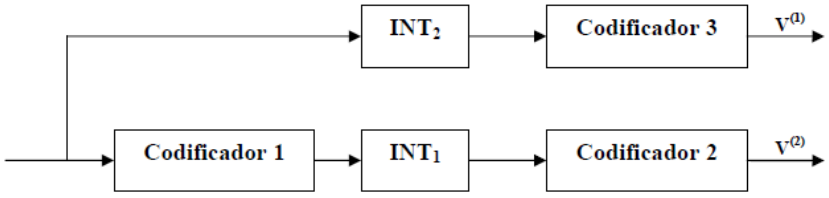

Figura 2.9. Arquitectura HCCC

Esta arquitectura no ha sido aún estudiada en profundidad. Pese a ello, sí se sabe que constituye un mejoramiento de la estructura SCCC.

Respecto a la decodificación, los turbo códigos se decodifican mediante decodificadores SISO (*Soft-In-Soft-Out*) no complejos que pueden emplear un algoritmo de decodificación de máxima probabilidad. De acuerdo con [Barbulescu 04], "utilizando decodificadores SISO la información puede ser pasada de un decodificador al siguiente en un modo iterativo. Esta es una estrategia de "divide y vencerás" que en un proceso iterativo vienen puede aproximarse al desempeño de los MLDA[5]". La estructura general de un turbo decodificador iterativo es la siguiente [Hanzo 2000]:

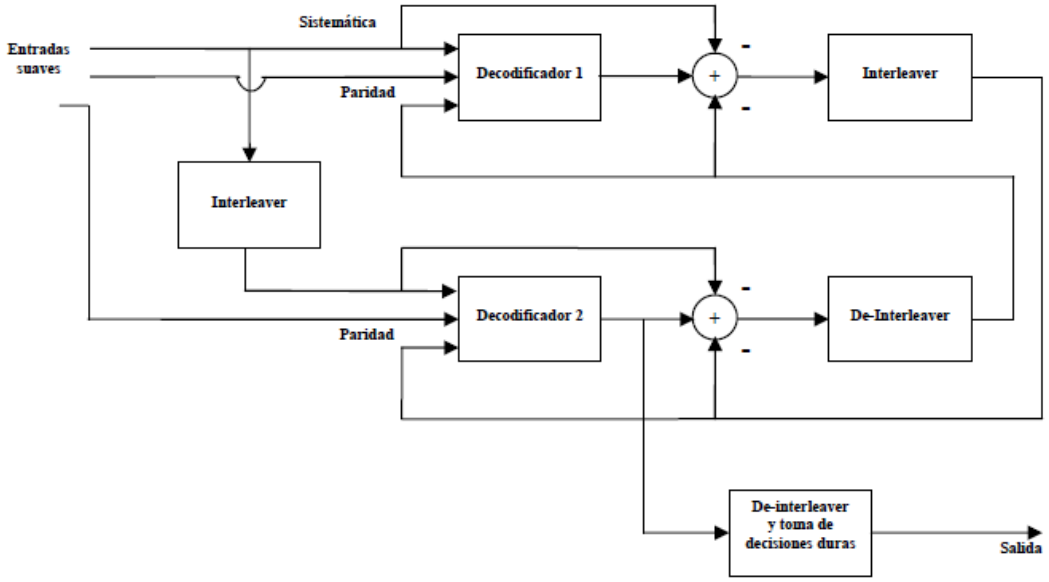

Figura 2.10. Arquitectura general de un turco decodificador

Esta figura corresponde a un turbo decodificador de un turbo codificador de tasa 1/3, y representa el tipo de decodificador iterativo utilizado más comúnmente. Cada decodificador tiene tres entradas: los bits de salida del canal codificados sistemáticamente, los bits de paridad generados por el codificador correspondiente en el Turbo-codificador, y la información proveniente del otro decodificador acerca de los valores probables de los bits decodificados (información a priori).

Los decodificadores tienen que entregar salidas *soft* valiéndose de las entradas provenientes del canal y de la información entregada por el otro decodificador. Estas salidas *soft* se representan generalmente mediante las *relaciones Log-likelihodd (LLR's*[6]), que dan información sobre el signo de cada bit y la probabilidad de una. La LLR para el valor de un bit decodificado u_k está dada por:

(2.6)

$$L(u_k) = ln\left(\frac{P(u_k = +1)}{P(u_k = -1)}\right)$$

Donde $P(u_k = +1)$ es la probabilidad de que el bit decodificado sea un "1" binario, mientras que $P(u_k = -1)$ representa la probabilidad de que se trate de un "0" binario. Una LLR positiva y alta indicará una probabilidad alta de que se trate de un 1 y una negativa (por tratarse de un logaritmo) y alta indicará una probabilidad alta de que se trate de un 0.

[5] MLDA: *Maximum Likelihood Decoding Algorithms* o Algoritmos de Decodificación de Máxima Probabilidad.
[6] LLR: Log-Likelihood Ratios

En la figura 2.10 también se observa que las entradas al turbo codificador son *soft*, por lo que será necesario que previo al turbo codificador haya un estimador de la LLR, que estime las relaciones *log-likelihodd* de los bits recibidos de la señal demodulada. Debido a ello, los decodificadoras deben trabajar con entradas y salidas *soft*, por lo que se hace necesaria la utilización de algoritmos de decodificación SISO. Existen dos algoritmos de decodificación SISO: los algoritmos MAP (*Maximum a posteriori*) o algoritmos que minimizan la probabilidad de error de símbolo; y los algoritmos SOVA (*Soft OutpuT*t Viterbi Algoritm) o algoritmos que minimizan la probabilidad de error de palabra o secuencia.

2.3 Multitrayecto y desvanecimientos

Al atravesar las señales radioeléctricas diversos medios durante su propagación, la perdida de propagación se convierte en una variable aleatoria al depender de las características físicas de los medios que atraviesa. Si suponemos invariable con el tiempo el valor medio de la potencia transmitida, la variabilidad de la pérdidas debido al medio implica que las pérdidas de la potencia sean variables. Se denomina potencia recibida nominal al valor medio de la potencia recibida, y toda pérdida de la potencia recibida con respecto a su valor nominal es lo que llamamos desvanecimiento. La diferencia entre la potencia nominal y la recibida se conoce como profundidad de desvanecimiento, y el tiempo que media entre la disminución y la recuperación del valor nominal se define como duración del desvanecimiento.

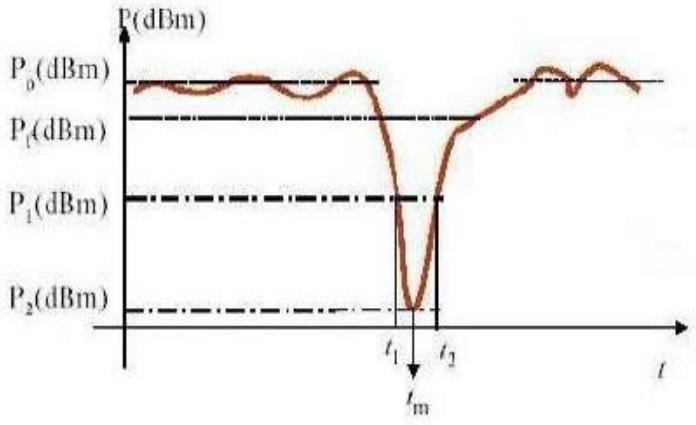

Figura 2.11.Desvanecimiento

En la figura se observa el valor nominal es P_0 y que se produce un desvanecimiento intenso, valiendo en su entorno la potencia media P_f que es inferior al nominal. En el instante t_1 se tiene una profundidad de desvanecimiento P_0 - P_1 y a continuación se llega al máximo desvanecimiento P_0 - P_2. Seguidamente el nivel se recupera volviendo a alcanzar el valor P_1, por lo tanto la duración de este desvanecimiento ha sido de t_2 - t_1.

Dentro de los desvanecimientos se puede hacer una primera gran división:

- Desvanecimientos planos: cuando la caída de nivel afecta por igual a todas las componentes del espectro de una portadora modulada,

- Desvanecimientos selectivos: cuando la caída de nivel afecta a una parte del espectro de la señal. Estos desvanecimientos introducen distorsión a la señal al afectar de modo diferente a unas frecuencias y a otras.

Definimos el multitrayecto como la llegada de varias versiones de la señal original debido a la propagación por múltiples caminos por las reflexiones producidas. Debido a esto en recepción nos encontraremos con una señal que será la suma de la señal original más las distintas versiones de la original. El multitrayecto produce desvanecimientos selectivos en frecuencia.

Para poder estudiar el efecto del desvanecimiento selectivo (el ocurrido debido al multitrayecto, por ejemplo) necesitamos conocer dos datos:

- El porcentaje de tiempo en que un desvanecimiento multitrayecto tendrá carácter selectivo.
- La FTM o lo que es lo mismo la función de transferencia multitrayecto, que no es más que un modelo de la función de transferencia del canal.

La FTM se va a estudiar mediante el modelo de rayos[7]. El modelo de rayo más general tiene en cuenta el rayo directo y *N* rayos de trayectos múltiples. La FTM para este modelo es de la forma [Rábanos 08]:

(2.7)

$$H(w) = \sum_{i=0}^{N} a_i \cdot e^{-j(\omega\tau_i+\varphi_i)}$$

Donde a_i, τ_i, φ_i son la amplitud, el retardo y la fase del i-ésimo rayo. Como estos parámetros son difíciles de medir, se propone un modelo normalizado [Rábanos 08]:

(2.8)

$$H(w) = H_0 + \sum_{i=0}^{N} a_i' \cdot e^{-j(\omega\tau_i'+\varphi_i')}$$

Donde H_0 es el valor de H(w) para el rayo directo y a_i', τ_i', φ_i' son las amplitudes, retardos y fases de los rayos, con referencia al rayo directo.

En la práctica, si tenemos N = 2 la ecuación anterior puede ajustarse a datos obtenidos de mediciones experimentales, tomando como referencia H_0 = 1.

2.4 Prefijo cíclico

Como se ha visto en el apartado 2.3 debido al multitrayecto al receptor llegan varias versiones del símbolo transmitido junto al original, lo que lleva a la pérdida de ortogonalidad y a la interferencia inter simbólica. Para ello se utiliza el prefijo cíclico. El prefijo cíclico no es más que repetir una parte del final del símbolo OFDM al principio del mismo. En la siguiente figura se muestra un ejemplo.

[7] La FTM también se puede estudiar mediante modelos polinómicos. Para más detalles consultar [Rábanos 08]

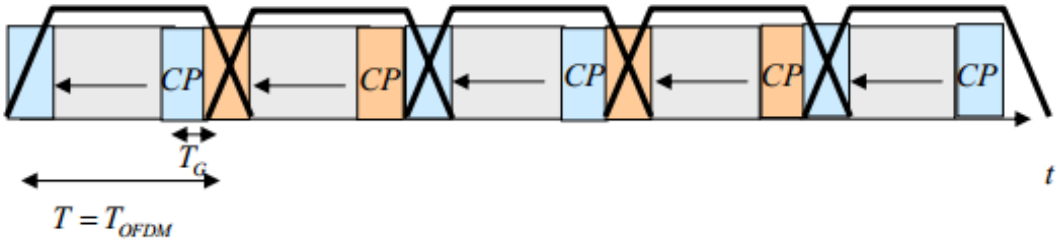

Figura 2.12. Prefijo cíclico

Debido a las réplicas la duración de la respuesta impulsiva del canal no es T_{OFDM}, pasando a ser igual a T_{OFDM}+T_R, siendo T_R el mayor de los retardos de las versiones del original. Para que el prefijo cíclico funciones, éste debe ser mayor o igual que la duración de T_R, es decir:

$$T_{CP} \geq T_R$$

Con la inserción del prefijo cíclico conseguimos que la integración de cada una de las réplicas se haga sobre el símbolo OFDM completo, asegurando que los productos de las subportadoras diferentes se integren en intervalos completos T_{OFDM}, con lo que se mantiene la ortogonalidad y se evita la interferencia entre ellas. Para que quede más claro lo explicado en este párrafo se muestra la figura 2.13 [AGUSTI 10]:

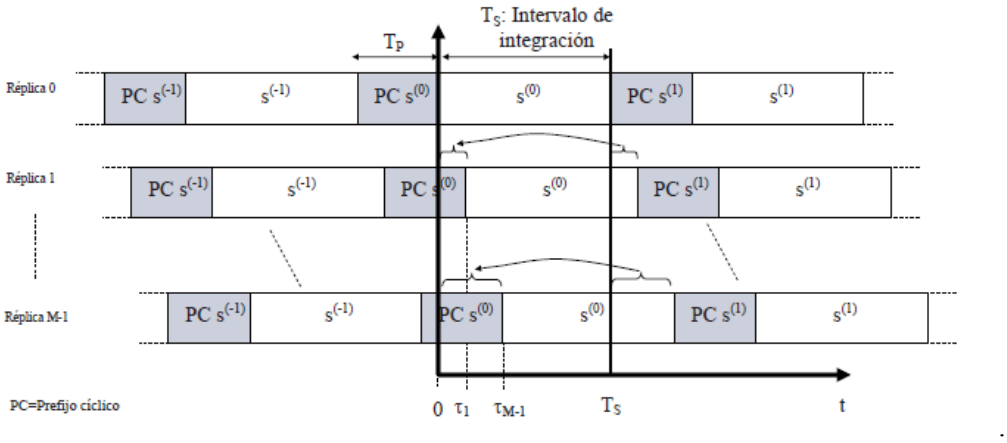

Figura 2.13. Periodo integración con prefijo cíclico.

En la figura se observa como en el intervalo de integración la señal integrada únicamente está formada por la deseada y réplicas de la misma. No sufre influencia de la señal anterior.

2.5 Ecualización

Como se ha dicho en el apartado 2.1.3, una de las ventajas que presenta la modulación OFDM es la posibilidad de ecualización en frecuencia, de manera más o menos sencilla, evitando la tradicional ecualización en el dominio del tiempo que resulta más compleja. Decimos que la ecualización es sencilla debido al hecho de trabajar con n portadoras en paralelo y poder ecualizar cada una por separado.

La base de la ecualización se basa en obtener la respuesta impulsiva del canal, lo cual es más sencillo en el dominio de la frecuencia. La idea es la que conocida la señal transmitida X[ω] y una vez que obtenemos la recibida Y[ω] podemos calcular la respuesta del canal en frecuencia:

(2.9)

$$y[n] = x[n] * h[n] \xrightarrow{TF} Y[\omega] = X[\omega]H[\omega] \longrightarrow H[w] = {}^{Y[w]}\!/_{X[w]}$$

Lo primero que hay que decir es que obtener la respuesta impulsiva exacta del canal es imposible, ya que al ser el ruido aleatorio e inherente a cualquier sistema de comunicaciones cada vez que se intente el resultado será algo diferente al anterior debido a que el ruido varía. Por lo tanto, la señal recibida z[ω] será la suma de señal deseada Y[ω] y del ruido R[ω], pudiendo calcular la respuesta aproximada del canal $H'[\omega]$:

(2.10)

$$z[n] = x[n] * h'[n] \xrightarrow{TF} Z[\omega] = X[\omega]H'[\omega] \longrightarrow H'[w] = {}^{Z[w]}\!/_{X[w]}$$

Esta respuesta del canal se puede obtener de distintas formas, por ejemplo mediante el uso de portadoras piloto, que no son más que portadoras que contienen información conocida por el receptor. El receptor sabe, obviamente, la posición de cada una en cada instante de tiempo.

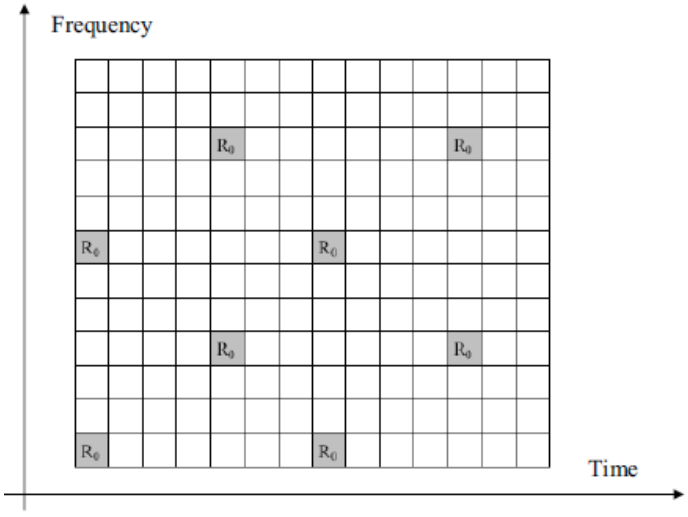

Figura 2.14. Portadoras piloto.

En la figura se observan las portadoras piloto están representadas por cuadros grises. Las portadoras pueden estar fijas en la misma posición siempre o ir variando, transmitirse en todas las ráfagas o no, etc. En este ejemplo concreto se trata de dos portadoras piloto, las cuales se encuentran en dos frecuencias distintas cada una que se van alternando y no se transmiten todo el tiempo, aparecen y desaparecen. Con las portadoras piloto se calcula una aproximación de la

respuesta impulsiva del canal con la que poder corregir las portadoras próximas que transmiten información útil.

Otra forma de obtener la respuesta del canal sería utilizar durante un instante de tiempo todo las subportadoras que forman el espectro o una gran parte de ellas con símbolos conocidos en recepción. Estas ráfagas sirven para estimar el canal. La idea es similar al uso de portadoras pilotos, sólo que en este caso se ocupa todo o gran parte del ancho de banda. La ventaja que presenta es que se obtiene una respuesta del canal para todas o gran parte de las subportadoras a cambio de no poder transmitir información durante el tiempo que dure el símbolo OFDM. Esta ráfaga habrá que repetirla cada cierto tiempo ya que en un caso real, por ejemplo, de radiodifusión el canal va variando con el tiempo.

Una vez que tenemos la respuesta del canal podemos modelar el ecualizador. Una manera de modelarlo es mediante el criterio MMSE (*Minimum Mean Square Error*). Este criterio se basa en minimizar el error cuadrático medio entre la transmisión de símbolo d_k y el estimado para la salida del ecualizador d'_k. En otras palabras los coeficientes W_k que multiplican a las subportadoras son escogidos para minimizar $E[d_k - d'_k]^2$.

2.6 Entrelazado

La idea del entrelazado es muy sencilla, consiste en una etapa que intercambia la posición de los bits antes de realizar la modulación QAM (después de ser codificados). Para comprender mejor la idea se presenta la siguiente figura:

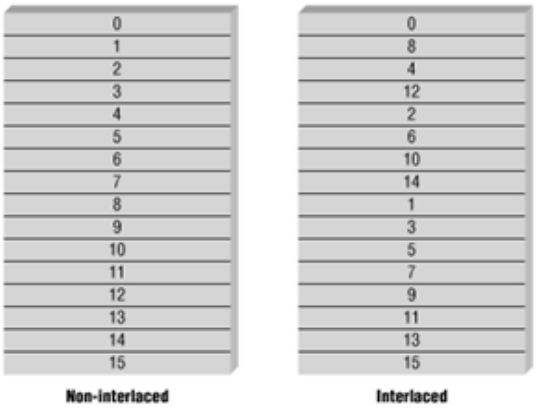

Figura 2.15. Entrelazador

Como hemos visto en el apartado 2.3 los desvanecimientos selectivos afectan de distinta manera a las subportadoras, debido a esto es posible que algunas queden muy afectadas mientras otras apenas han sufrido variación. Si este ocurre la información de esa portadora queda totalmente irrecuperable pese a que haya habido codificación. Debido a ello se realiza un entrelazado para que los bits consecutivos sean transmitidos en distintas subportadoras y así si un símbolo entero de información transmitido en una subportadoras queda inservible, al desentrelazar los errores se repartan y permite que el codificador los corrija.

2.7 Método de Montecarlo

El método de Montecarlo es un método cuasi-analítico, conceptualmente simple y el más utilizado para calcular la BER (tasa de bit erróneo) y la SER (tasa de símbolo erróneo). El método para el caso de medir la BER y la SER consiste en comparar los símbolos o bits recibidos con los transmitidos para calcular la posibilidad de que un símbolo sea erróneo P_e o un bit P_b Glover 04]:

(2.11)

$$P_e = \frac{Número\ de\ errores\ contabilizados}{Número\ de\ símbolos\ transmitdos}$$

Por lo que la SER sería simplemente [Glober 04]:

(2.12)

$$SER = P_e R_s \text{ Error/s}$$

Siendo R_s la tasa de símbolos en baudios. En nuestro caso calculamos únicamente la BER por lo por lo que lo que trabajaríamos con P_b.

Hasta ahora no estamos más que aplicando la definición de probabilidad más elemental. El problema del método de Montecarlo es que para tasas de error bajas si queremos obtener precisión y fiabilidad en las tasas obtenidas debemos gran cantidad de simulaciones. Y a mayor tasa de fiabilidad deseada más simulaciones deberemos realizar. Para hacernos una idea de la forma en que varían las tasas de fiabilidad de la BER obtenida en función del número de bits observados, mostramos la siguiente figura [Jeruchim 84]:

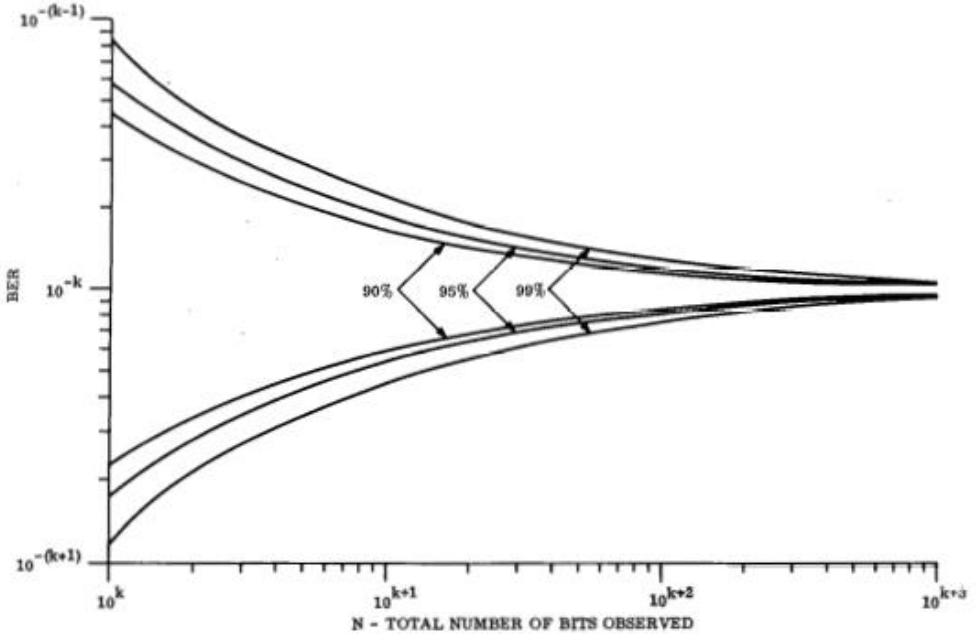

Figura 2.16. Variación de la BER en función de la fiabilidad.

Observamos como dependiendo del margen de fiabilidad empleado necesitaremos obtener más o menos bits erróneos para una determinada BER. Es decir, que si queremos obtener un margen de

fiabilidad mayor necesitaremos obtener más bits erróneos para calcular un mismo valor de BER, que con un margen de error menor.

CAPÍTULO 3

TEORÍA ORIENTADA A LA SIMULACIÓN

En este capítulo se van analizar los conceptos teóricos necesarios para poder comprender los bloques que conforman el esquema de nuestro sistema OFDM.

3.1 OFDM (modulación y demodulación)

Para llegar al esquema del modulador y demodulador OFDM empleado en este proyecto, primero debemos desarrollar la señal OFDM muestreada para obtener su versión discreta. Para ello partimos de la ecuación (2.3):

$$s(t) = \sum_{k=0}^{K-1} d_k e^{j2\pi k\Delta f t} \cdot rect_{T_s}$$

Si muestreamos la señal s(t) con N muestras y un periodo de muestreo $T_m = \frac{1}{f_m} = \frac{1}{N\Delta f}$ obtenemos la señal discreta s[n]:

(3.1)

$$s[n] = \sum_{k=0}^{K-1} d_k e^{j2\pi k\Delta f n T_m} \cdot rect_{T_s}[nT_m] = \sum_{k=0}^{K-1} d_k e^{j2\pi kn/N} \cdot rect_{T_s}$$

Si comparamos la señal s[n] con la definición de la Transformada discreta de Fourier (DFT) y la inversa de la Transformada discreta de Fourier (IDFT):

(3.2)

$$\text{DFT: } X[k] = \sum_{n=0}^{N-1} x[n]\, e^{-j\frac{2\pi}{N}kn}$$

(3.3)

$$\text{IDFT: } x[n] = \frac{1}{N}\sum_{n=0}^{N-1} X[k]\, e^{j\frac{2\pi}{N}kn}$$

Comparando la ecuación (3.1) con la (3.3) se observa que el símbolo OFDM no es más que la IDFT de cada símbolo d_k multiplicado por una constante, o lo que es lo mismo:

(3.4)

$$s[n] = N \cdot IDFT(d_k)$$

Y observando la ecuación (3.2) llegamos a la conclusión de que para recuperar el símbolo debemos realizar:

(3.5)

$$d_k = \frac{1}{N} \cdot DFT(s[n])$$

Por lo tanto para obtener nuestro símbolo OFDM basta con realizar la IDFT (IFFT[8] en concreto) a cada símbolo d_k. Por lo tanto el esquema implementado como modulador OFDM es el siguiente:

[8] IFFT: Transformada Rápida de Fourier Inversa. Algoritmo eficiente para calcular la IDFT

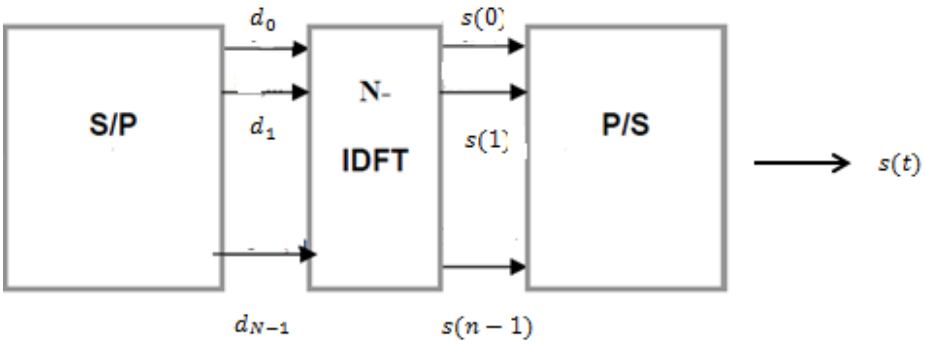

Figura 3.1. Esquema modulador OFDM implementado

Y análogamente el esquema del demodulador consistirá en emplear la DFT(FFT[9]) y será:

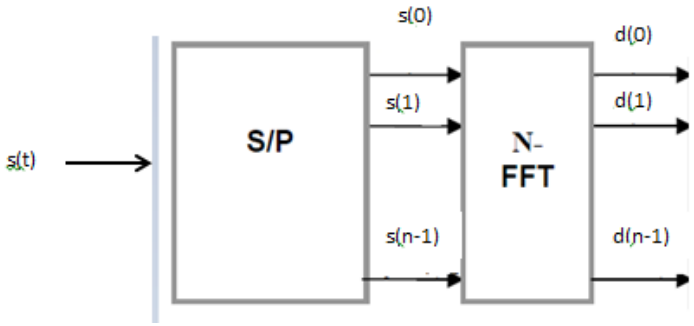

Figura 3.2. Esquema demodulador OFDM implementado

3.2 Codificación

A continuación se muestran los esquemas de codificación empleada:

3.2.1 Código convolucional

El codificador convolucional empleado para la simulación es el definido para el estándar UMTS y conservado para LTE [LTE 2011]:

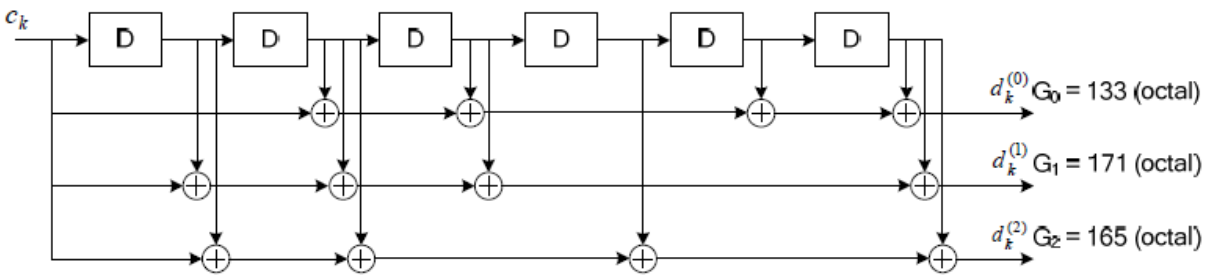

Figura 3.3. Esquema código convolucional empleado

[9] FFT: Transformada Rápida de Fourier. Algoritmo eficiente para calcular la DFT

Se trata de un codificador convolucional definido por los siguientes parámetros $n = 3$, $k = 1$, $m = 6$ y $L = 5$. Por lo tanto la tasa de este codificador es de 1/3.

3.2.2 Turbo código

El turbo código empleado en este proyecto es el utilizado en el estándar CDMA 2000. Su esquema es [CDMA2000 2009]:

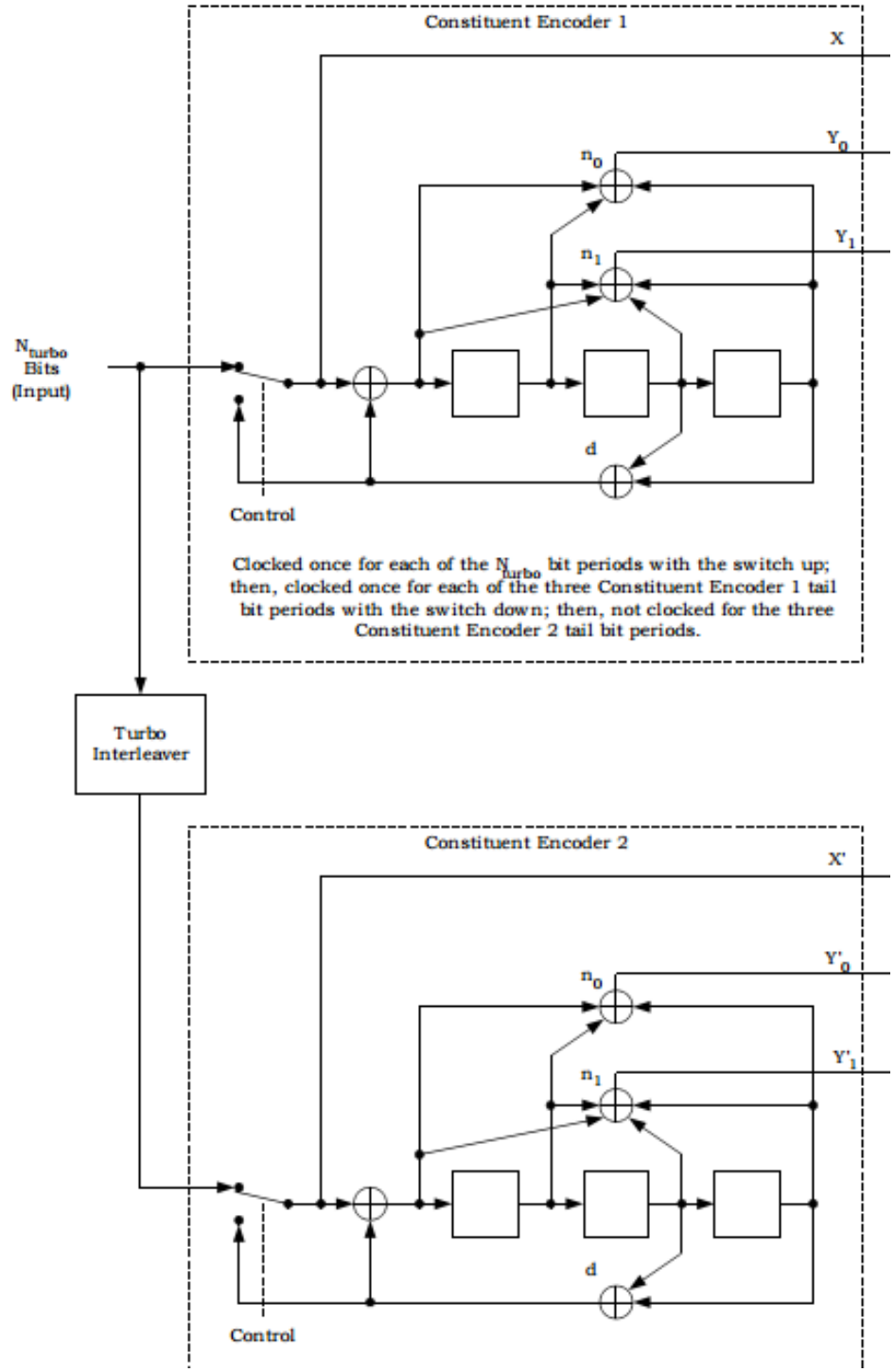

Figura 3.4. Esquema turbo código empleado

Hay que decir que la salida sistemática del segundo codificador (x') no se utiliza. En la figura 3.4 se muestra el mismo esquema pero con un esquema más claro y con las salidas que únicamente se utilizan:

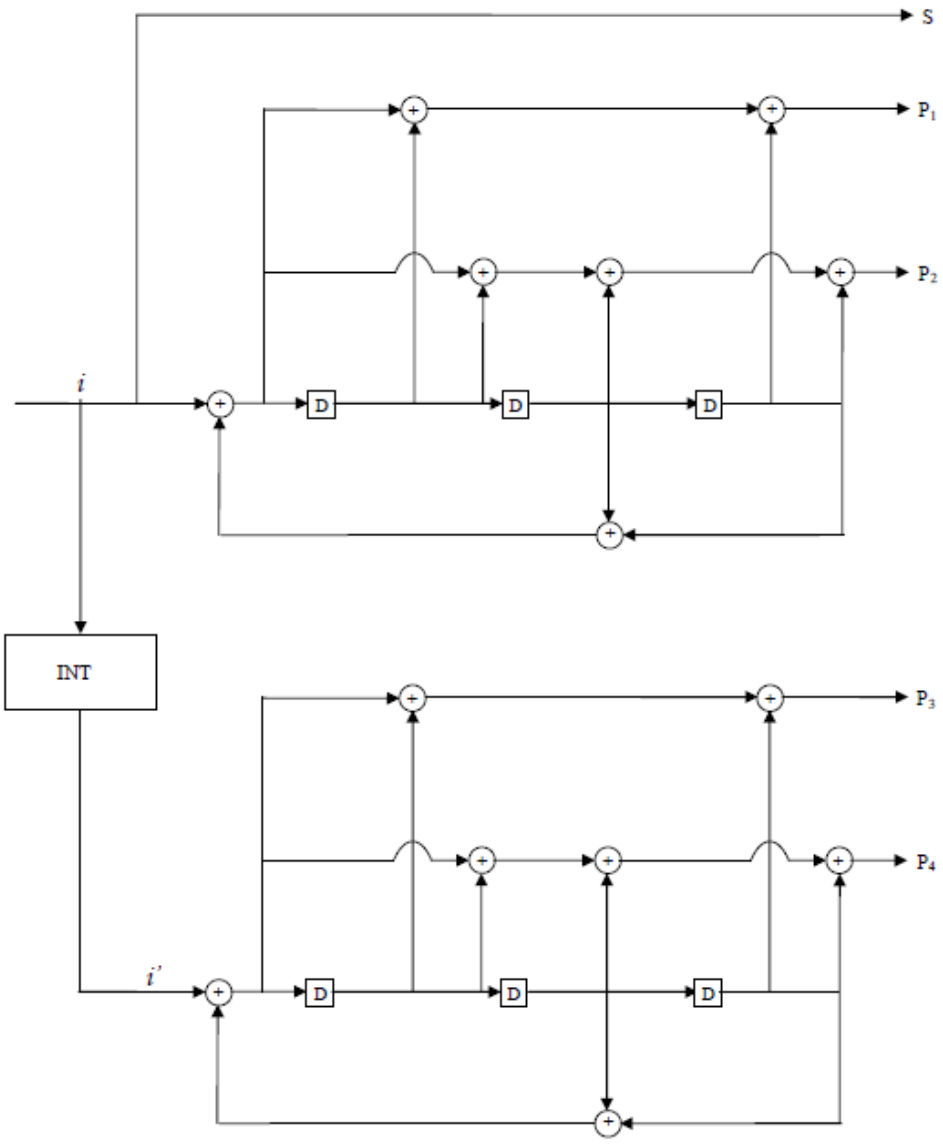

Figura 3.5. Esquema turbo código empleado simplificado

Se trata de un turbo código paralelo (PCCC) implementado mediante dos codificadores RSC idénticos. Presenta cinco salidas: la sistemática y dos cada codificador convolucional. El *interleaver* usado para la simulación es una permutación pseudoaletoria conocida tanto en el transmisor como en el receptor. Para la decodificación se utiliza el algoritmo *MAP*.

3.3 Ecualizador

El ecualizador utilizado se trata de un ecualizador MMSE. La ecualización se realiza en el dominio de la frecuencia y su funcionamiento se basa es mandar una primera ráfaga de entrenamiento (conocida por el transmisor la información mandada) para estimar la respuesta del canal. Como se dijo en el capítulo 2 nunca se podrá calcular la respuesta impulsiva exacta debido al ruido.

Una vez obtenida la aproximación de la respuesta del canal $H[\omega]$, se puede obtener los coeficientes del ecualizador por los que multiplicar cada portadora. Si no hubiera ruido, está respuesta sería siempre la misma y por lo tanto se podrían corregir todos los errores debido, por ejemplo, al multitrayecto, ya que bastaría con dividir la señal recibida entre la respuesta del canal para contrarrestar los efectos del medio. Debido al ruido esto no es posible por lo que hay que buscar de otra manera los coeficientes por los que multiplicar la señal recibida para contrarestar los efectos del canal. Estos coeficientes se calculan buscando el mínimo error cuadrático medio. De esta manera el coeficiente W_k por el que habría que multiplicar cada portadora sería [Sesia 09]:

(3.6)

$$W_k = \frac{H_k^*}{|H_k|^2 + N_0}$$

Siendo H_k la respuesta del canal para cada portadora obtenida con la ráfaga de entrenamiento y N_0 la potencia de ruido.

3.4 BER

3.4.1 QAM

Ya que una señal M-QAM es equivalente a dos señales PAM [10] en cuadratura, la probabilidad de símbolo erróneo P_M de una modulación M-QAM puede expresarse a partir de la probabilidad de símbolo erróneo $P_{\sqrt{M}}$ de una modulación $\sqrt{M}$-PAM. Concretamente P_M valdría [Proakis 01]:

(3.7)

$$P_M = 1 - (1 - P_{\sqrt{M}})^2$$

Siendo $P_{\sqrt{M}}$

(3.8)

$$P_{\sqrt{M}} = 2\left(1 - \frac{1}{\sqrt{M}}\right) Q\left(\sqrt{\frac{3}{M-1}\frac{kE_b}{N_0}}\right)$$

Hay que decir que esta expresión es válida únicamente para modulaciones M-QAM cuya constelación sea cuadrada. Para las modulaciones M-QAM no cuadradas (M= 2^k, k impar) es difícil obtener la tasa de símbolo erróneo ya que la constelación puede configurarse de distintas maneras. Una aproximación generalmente aceptada es:

[10] PAM. Modulación por Amplitud de Pulsos.

(3.9)

$$P_M \leq 4Q\left(\sqrt{\frac{3kE_b}{(M-1)N_0}}\right)$$

3.4.2 Códigos convoluciones

Respecto a los códigos convoluciones, no existe ninguna ecuación para calcular la probabilidad de bit erróneo en función de la potencia de ruido, únicamente existen aproximaciones para calcular los límites superiores e inferiores en los que se movería esa probabilidad. Para llegar a esos límites vamos a empezar definiendo otro concepto: la probabilidad de que una palabra de *k* bits de entrada al codificador sea incorrectamente decodificada

Primero estudiamos el caso de una decodificación tipo *soft* [Artés 07]:

(3.10)

$$P_e \approx k_2 Q\left(\sqrt{\frac{2D_{min}E_s}{N_0}}\right) = k_2 Q\left(\sqrt{\frac{2kD_{min}}{n}}\sqrt{\frac{E_b}{N_0}}\right)$$

Donde D_{min} es la distancia mínima de una constelación, $E_s$la energía media por símbolo y k_2[11] el número medio de errores que provoca en el decodificador el evento D_{min}, siendo D_{min} la distancia mínima de una constelación:

(3.11)

$$D_{min} = \min\{d(a_i, a_j)\}$$

Los valores de *i y de j* van desde 0 hasta M-1 siendo $i \neq j$.

Para realizar el mismo cálculo para una decodificación *hard* debemos sustituir la función Q en la ecuación (3.9) por la probabilidad de cometer más de un $(D_{min}-1)/2$ errores en un número *nz* de bits consecutivos, donde *z* es la longitud del evento erróneo D_{min}. Realizando esta sustitución obtenemos [Artés 07];

(3.12)

$$P_e \approx k_2 \sum_{i=[(D_{min}-1)/2]+1}^{nz} \binom{nz}{i} p^i (1-p)^{nz-i}$$

Donde p es la probabilidad del cruce del BSC (Canal Binario Símétrico).

[11] Para más información sobre como calcular k_2 consultar página 324 [Artés 07].

Una vez obtenida P_e podemos acotar la BER teniendo en cuenta que un error de decodificación se traducirá en entre 1 y k bits erróneos [Artés07]:

(3.13)

$$\frac{1}{k}P_e \leq BER \leq P_e$$

3.4.3 Turbo códigos

Respecto a los turbo códigos no es posible dar una definición para la probabilidad de su error de palabra. A modo de ejemplo se muestra a continuación un esquema de un turbo codificador de tasa 1/2 [Berrou 93]

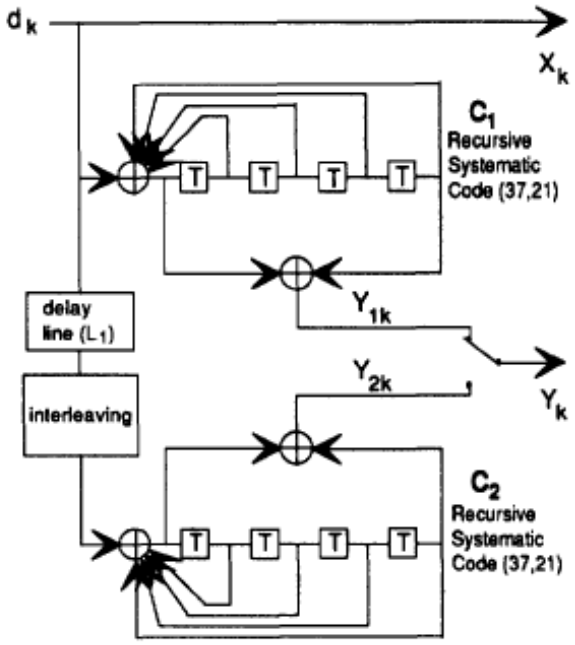

Figura 3.6. Esquema codificador [Berreu 93]

Y la BER obtenida en función del número de iteraciones mediante simulación utilizando el método de Montecarlo:

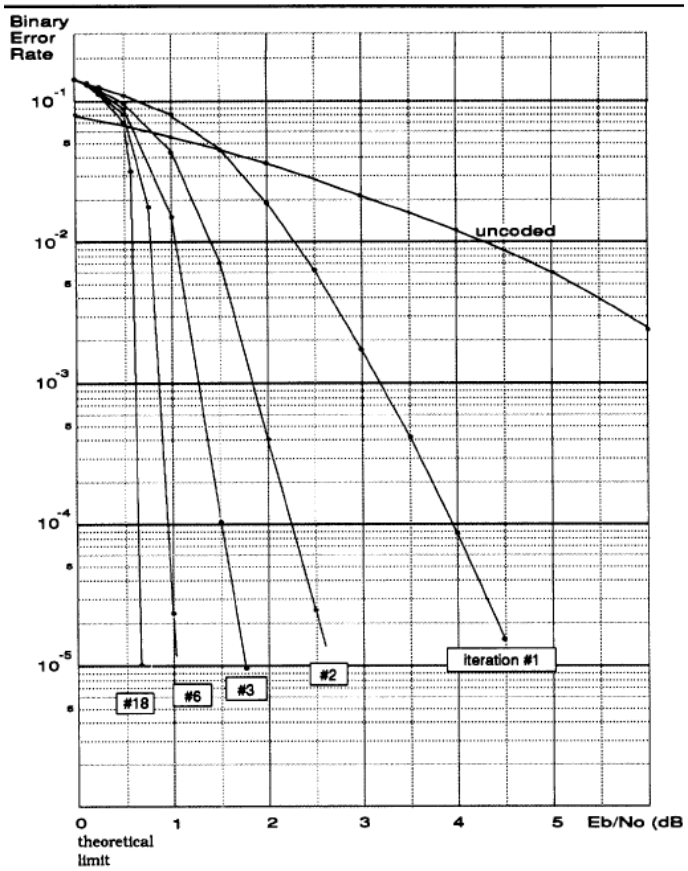

Figura 3.7. BER [Berreu 93]

En la gráfica se observan valores no vistos antes para Eb/No muy bajos. Comprobando que con el mismo sistema a base de subir el número de iteraciones el nivel Eb/No desciende enormemente.

3.5. Multitrayecto

El esquema de multitrayecto implementado está basado en el modelo de rayos. Concretamente es un canal multitrayecto discreto invariante en el tiempo (la respuesta impulsiva del canal no cambia), donde las atenuaciones son únicamente reales y constantes (al no depender del tiempo).

En nuestro esquema al receptor llega el rayo original sin atenuar ni sufrir retardo y los demás rayos provenientes del multicamino, estando cada uno atenuado $\tilde{a}_k$ y retardo τ_k. Esos valores de atenuación y de retardo son los que se introducen en el programa.

En la siguiente figura se muestra el esquema del sistema implementado:

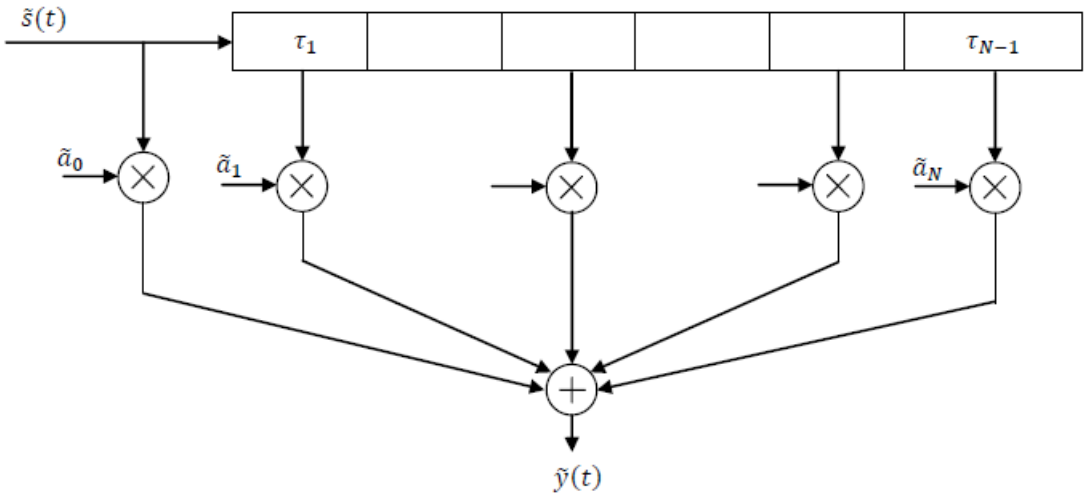

Figura 3.8. Esquema Multitrayecto implementado

En la figura 3.8 $\tilde{s}(t)$ es la el símbolo OFDM transmitido, e $\tilde{y}(t)$es la señal captada en el receptor que no es más que la suma de la transmitida más las distintas versiones de la señal original atenuadas y retardadas

3.6 Esquema del sistema implementado

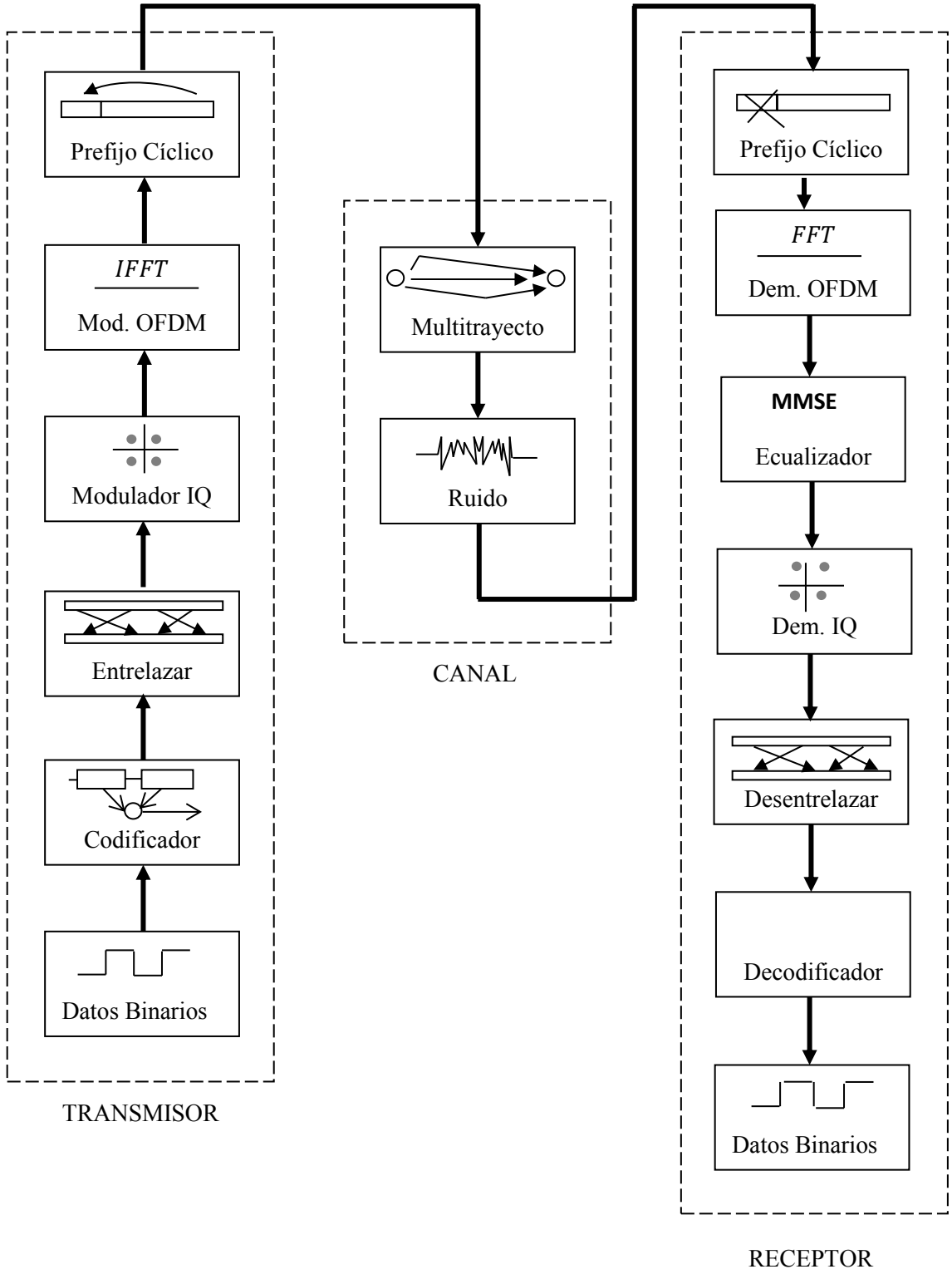

Figura 3.9. Esquema completo del sistema OFDM implementado

CAPÍTULO 4

IMPLEMENTACIONES Y MANUAL DE USUARIO

En este capítulo se explica el esquema OFDM desarrollado para un mejor entendimiento de la simulación de nuestro sistema, así como las funciones de Matlab utilizadas para dicha simulación. En la segunda parte del capítulo se elabora un manual de usuario para el correcto conocimiento y usa de la GUI (Graphical User Interface) implementada en este proyecto.

4.1 Funciones y subprogramas

Para poder realizar la simulación se han desarrollado funciones (principales y auxiliares o secundarias) y subprogramas.

4.1.1. Funciones principales

Son las funciones primordiales encargadas de simular todos los bloques del sistema implementado: codificador, modulador, generador de bits…

1. **BIN**:

```
function [ Datos_Binarios ] = BIN(n)
```

Genera una secuencia aleatoria con la cantidad de bits necesarios según la codificación, el tipo de QAM seleccionado, y el número de portadoras..

Parámetro de entrada:

- `n:` Tamaño de la secuencia (número de bits).

Parámetro de salida:

- `Datos_Binarios:` bits de información.

2. **Codificador:**

```
function [ Datos_Codificados ] =
Codificador(tipo,Datos_Binarios,intrlvrIndices)
```

Dependiendo de la opción seleccionada codifica los bits de información mediante un codificador convolucional o mediante un turbo código.

Parámetros de entrada:

- `tipo`: 1 para la codificación convolucional y 2 para la turbo codificación.
- `Datos_Binarios`: bits de información que hay que codificar.
- `intrlvrIndices`: *interleaver* utilizado por el turbo codificador.

Parámetro de salida:

- `Datos_Codificados`: bits de información codificados.

3. **Decodificador:**

```
function [ Bits_decodificados ] =
Decodificador(Datos_Codificados,tipo,numIter,intrlvrIndices,
HoS)
```

Decodifica los bits recibidos.

Parámetros de entrada:

- `Datos_Codificados:` datos a decodificar.

- `Tipo: 1 para decodificar un código convolucional y 2 para un turbo código.`
- `numIter:` número de iteraciones a utilizar en la decodificación de un turbo código.
- `intrlvrIndices`: *interleaver* utilizado en la turbo codificación. Es necesario saberlo para poder decodificar un turbo código correctamente.
- `HoS:` indica el modo de realizarse la decodificación de un código convolucional: 1 para *hard*, 2 para *soft*.

Parámetro de salida:

- `Bits_decodificados:` bits de información recuperados en recepción.

4. Entrelazar

```
function [ Datos_Entrelazados ] =                    Entrelazar(
Datos_Codificados, permutacion2 )
```

Intercambia el orden de los bits.

Parámetros de entrada:

- `Datos_Codificados:` bits a entrelazar.
- `permutacion2:` indica la nueva posición de cada bit.

Parámetro de salida:

- `Datos_Entrelazados:` bits entrelazados.

5. Desentrelazar

```
function [ Datos_Desentrelazados ] =             Desentrelazar(
Datos_SinIQ_RX, permutacion2 )
```

Pone los bits en su posición original.

Parámetros de entrada:

- `Datos_SinIQ_RX:`datos entregados por el demodulador QAM.
- `permutacion2:` indica la manera en que se han cambiado las posiciones de los datos al entrelazar.

Parámetro de salida:

- `Datos_Desentrelazados:` bits en sus posiciones originales.

6. ModIQ

```
function [ Datos_IQ_TX ] = ModIQ(Tipo_QAM,Datos_Modular,Nc)
```

Realiza la modulación NQAM de cadenas de bits.

Parámetros de entrada:

- `Tipo_QAM:` tipo de QAM: 4,16…
- `Datos_Modular`: datos a modular.
- `Nc:` número de portadoras

Parámetro de salida:

- `Datos_IQ_TX:` datos modulados.

7. DemIQ

```
function [ Datos_sinIQ ] =
DemIQ(Datos_IQ_RX,Tipo_QAM,ceros_ayuda)
```

Realiza la demodulación NQAM tras realizar la FFT al símbolo OFDM recibido.

Parámetros de entrada:

- `Datos_IQ_RX:` símbolos NQAM a demodular.
- `Tipo_QAM:` tipo de QAM: 4,16…
- `Ceros_ayuda`: indica los ceros añadidos (si ha sido necesario) para poder modular correctamente cada portadora en NQAM. Si han sido añadidos se quitarán en esta función.

Parámetro de salida:

- `Datos_sinIQ:` datos demodulados. Volvemos a tener unos y ceros.

8. OFDM_TX

```
function [ Datos_IFFT_TX ] = OFDM_TX( Datos_IQ , Resolucion )
```

Realiza la IFFT de los las portadoras modulada en NQAM para obtener el símbolo OFDM a transmitir.

Parámetros de entrada:

- `Datos_IQ:` portadoras moduladas en NQAM.
- Resolución: resolución utilizada para realizar la IFFT (Transformada de Fourier Rápida Inversa). Por defecto la resolución es igual al número de portadoras.

Parámetro de salida:

- `Datos_IFFT_TX:` símbolo OFDM a transmitir.

9. OFDM_RX

```
function [ Datos_FFT_RX ] = OFDM_RX( OFDM_Recibido, Resolucion)
```

Realiza la FFT del símbolo OFDM recibido.

Parámetros de entrada:

- `OFDM_Recibido:` símbolo OFDM recibido.
- Resolución: resolución utilizada para realizar la FFT (Transformada de Fourier Rápida) y obtener las portadoras moduladas en NQAM recibidas. Por defecto la resolución es igual al número de portadoras.

Parámetro de salida:

- `Datos_FFT_RX:` símbolo NQAM recibido.

10. prefijo_ciclico

```
function [ simbolo_OFDM_pref ] = prefijo_ciclico( simbolo_OFDM
, prefijo , NS , InOut)
```

Añade (en transmisión) o elimina (en recepción) el prefijo cíclico.

Parámetros de entrada:

- `simbolo_OFDM`: símbolo OFDM al que hay que añadir (símbolo OFDM transmitido) o eliminar (símbolo OFDM recibido) el prefijo cíclico.
- `prefijo`: indica el prefijo cíclico utilizado expresado en forma inversa (ya que la función lo utiliza dividiendo). Por ejemplo, para un prefijo de un 1/32, habrá que llamar a la función con el parámetro `prefijo` = 32.
- `NS`: número de muestras del símbolo OFDM.
- `InOut`: indica si se añade o elimina el prefijo cíclico. 1 para añadir el prefijo y 2 para eliminarlo.

Parámetro de salida:

- `simbolo_OFDM_pref`: símbolo OFDM con el prefijo añadido o eliminado.

11. multitrayecto

```
function [ MultitrayectoTime ] = multitrayecto( SimboloOFDM_TX
, Retardo , ATENUACION )
```

Dado un valor de atenuación y de retardo se calcula la réplica del símbolo OFDM original a transmitir. Es importante decir que únicamente devuelve un único multitrayecto, es decir, si se desea una señal que esté formada por el símbolo OFDM original y tres réplicas (con sus respectivas atenuaciones y retardos), se deberá llamar a esta función tres veces para calcular cada réplica.

Parámetros de entrada:

- `SimboloOFDM_TX`: símbolo OFDM a transmitir del que se desea calcular un multitrayecto.
- `retardo`: retardo para calcular la réplica.
- `ATENUACION`: atenuación para calcular la réplica.

Parámetro de salida:

- `MultitrayectoTime`: réplica obtenida a partir del símbolo OFDM original con su atenuación y retardo correspondiente.

12. Estimacion_camal

```
function [ respuesta_canal_Pn ] =
Estimacion_Canal(Nc,NQAM,SNR,Multi,aten,retar,NroTrayectos)
```

Calcula una aproximación[12] de la respuesta impulsiva del canal y la potencia de ruido. Para calcularlo manda un símbolo OFDM de entrenamiento con bits conocidos tanto en recepción

[12] Como se dijo en tema 3, debido al ruido es imposible calcular la respuesta impulsiva exacta del canal.

como en transmisión. Ni se añade prefijo cíclico, ni se codifica. En este caso, el esquema en recepción queda simplificado a un único bloque: el demodulador OFDM, ya que lo que buscamos es conocer la respuesta en frecuencia del canal, nada más.

Parámetros de entrada:

- `Nc:` número de portadoras utilizadas.
- `NQAM`: tipo de modulación de QAM utilizada.
- `SNR:` relación señal a ruido utilizada.
- `Multi:` si hay multitrayecto (`Multi = 1`) o no (`Multi = 2`).
- `aten:` es una matriz con la información de las atenuaciones de todos los multirayectos que se van a utilizar para simular el medio. Por ejemplo, si se desean 3 multitrayectos (o réplicas), `aten` será una matriz de 1x3.
- `retar:` es una matriz con la información de los retardos de todos los multirayectos que se van a utilizar para simular el medio.
- `NroTrayectos:` número de multitrayectos o réplicas que se utilizan para simular el medio. `NroTrayectos` no incluye al símbolo OFDM original, es decir, si deseamos simular que en recepción llega el original más tres réplicas, `NroTrayectos` valdrá 3.

Parámetro de salida:

- `respuesta_canal_Pn:` matriz de dimensión 1x(Nc+1) que en sus Nc primeras posiciones contiene una aproximación de la respuesta impulsiva del canal, y en la posición Nc+1 (última posición de la matriz) la potencia de ruido.

4.1.2. Funciones secundarias

Son funciones que son llamadas por las funciones principales o por otras funciones secundarias.

1. **relleno**

```
function [ add_zeros ] = Relleno( Datos_entrada_modulador
,Tipo_QAM )
```

Debido a la codificación puede ocurrir que no haya datos binarios suficientes para poder modular el último símbolo NQAM, en este caso se rellena con ceros. Esta función es llamada dentro de la función ModIQ.

Parámetros de entrada:

- `Datos_entrada_modulador:` datos a la entrada del modulador QAM. Interesa el tamaño de estos datos.
- Tipo_QAM: Tipo de modulación QAM que se va a realizar
- `Nc:` número de portadoras

Parámetros de salida:

- `add_zeros:` número de ceros que hay que añadir para realizar correctamente la modulación QAM.

2. **Relleno_OFDM**

```
function [datos_IQ_TX_total] = Relleno_OFDM(datos_IQ_TX,Nc)
```

Añade ceros a la entrada del modulador OFDM para completar la cadena en caso de que lo requiera. Para ello compara el número de símbolos a la entrada y el número de portadoras utilizada. Esta función es llamada por la función `OFDM_TX_graf`

Parámetros de entrada:

- `datos_IQ_TX:` datos a la entrada del modulador OFDM.
- `Nc:` número de portadoras utilizadas.

Parámetro de salida:

- `datos_IQ_TX_total:` datos a la entrada del modulador OFDM con la cantidad de ceros añadidos correspondientes.

3. OFDM_TX_graf

```
function [Datos_IFFT_TX_graf] = OFDM_TX_graf (Datos_IQ_TX , Nc , resolucion)
```

Realiza la IFFT de los las portadoras modulada en NQAM. Esta función se utiliza para representar el símbolo OFDM en frecuencia. Se diferencia de OFDM_TX en esa función siempre se utiliza al ser llamada como resolución el número de portadoras, mientras que aquí no.

Parámetros de entrada:

- `Datos_IQ_TX:` datos a la entrada del modulador OFDM.
- `Nc:` número de portadoras utilizadas.
- `Resolución:` resolución utilizada para la IFFT.

4.1.3. Subprogramas

Hay dos subprogramas principales que se valen de las funciones descritas en el apartado anterior para simular el sistema.

1. calcula.m

Es el subprograma principal encargado de simular todo el sistema con los valores seleccionados en la GUI. Para ellos se vale de todas las funciones descritas anteriormente y de la función propia del *toolbox* de Matlab *awgn* para añadir el ruido y poder simular completamente el sistema. Según las opciones seleccionadas en la GUI se simulará el sistema completo descrito en el tema 3, o una parte de él. En las figura 4.1, 4.2 y 4.3, se muestran los esquemas de transmisión, del canal y de recepción condicionados según las opciones seleccionadas en la GUI.

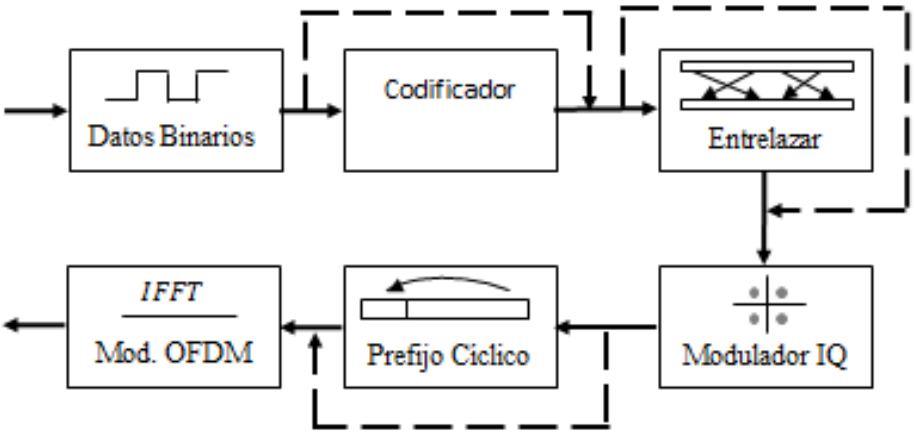

Figura 4.1. Esquema de Transmisión del sistema con bloques condicionados

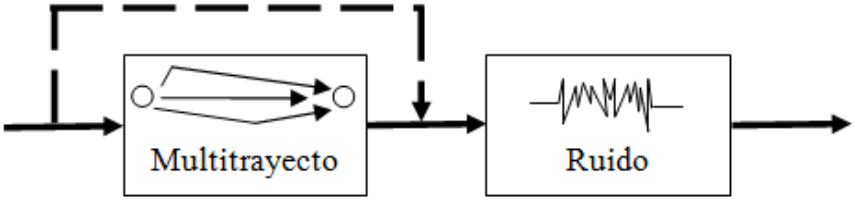

Figura 4.2. Esquema del canal del sistema con bloques condicionados

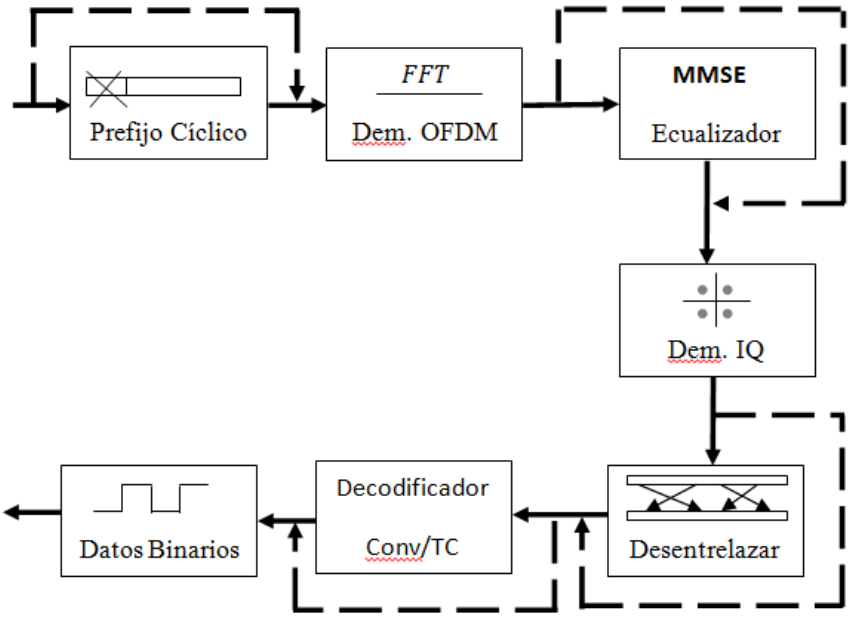

Figura 4.3. Esquema de recepción del sistema con bloques condicionados

2. `Estimacion_Canal.m`

Pese a que se trata de una función ya explicada en el apartado 4.1.1. Dentro de ella se desarrolla un versión simplificada de calcula.m (dentro de esta función no se llama a `calcula.m`) en las siguientes figuras se muestra el esquema utilizado para el cálculo de la respuesta del canal.

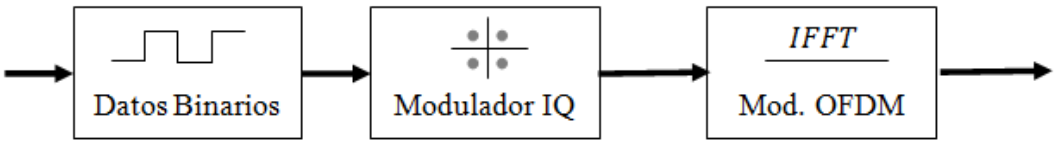

Figura 4.4. Esquema de transmisión utilizado por Estimacion_Canal

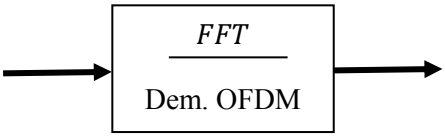

Figura 4.5 Esquema de recepción utilizado por Estimacion_Canal[13]

El esquema del canal utilizado por Estimacion_Canal es idéntico al de la figura 4.2.

3. `montecarlo.m`

Es el subprograma encargado de realizar las iteraciones (simulaciones) necesarias para calcular la BER. Para calcular cada punto de la BER realiza tantas iteraciones como sean necesarias basándose en dos parámetros indicados en la GUI:

- Número máximo de errores contabilizados.
- Número máximo de experimentos.

Las condiciones (la codificación utilizada, si hay multitrayecto, etc) de todas las iteraciones para cada Eb/No son definidas una única vez mediante la GUI. Una vez definidas estas condiciones se llama al subprograma `calcula.m` tantas veces como sea necesario hasta que se llega al número máximo de errores contabilizados definido, o hasta que se iguala el número máximo de experimentos indicado en la GUI. Este algoritmo se repite para todos los valores de Eb/No en los que se desea calcular la BER. Una vez que se han realizado todos los experimentos, tenemos contabilizados el número de bits erróneos contabilizados y el número de bits de información totales que se han utilizado, por lo que se puede calcular la BER.

[13] Para calcular la respuesta del canal únicamente necesitamos comparar los datos en el dominio de la frecuencia recibidos y transmitidos, por lo que con implementar en recepción el demodulador OFDM es suficiente.

La figura 4.6 muestra el diagrama de flujo del flujo para un nivel de Eb/No concreto. Habrá que repetir el diagrama para todos los niveles de Eb/No para los cuales se desee calcular la BER.

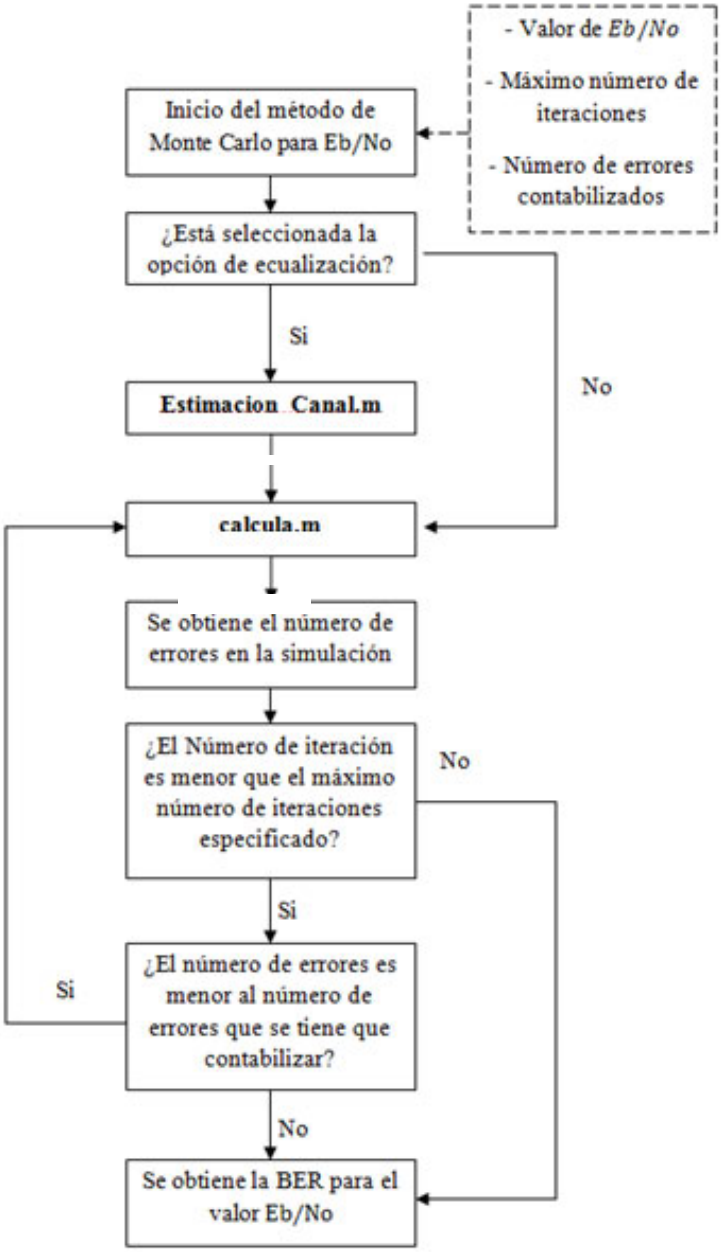

Figura 4.6. Esquema del Algoritmo para obtener la BER mediante el método de Monte Carlo[14]

[14] Cuando se llama a la función `Estimacion_Canal`, sólo se llama para obtener la potencia de ruido, no para obtener la respuesta del canal. Eso ha sido obtenido anteriormente y se obtiene una única vez para todos los valores Eb/N0.

4.2 Manual de usuario

Introduciendo el comando Inicio_GUI en Matlab nos aparecerá la siguiente interfaz:

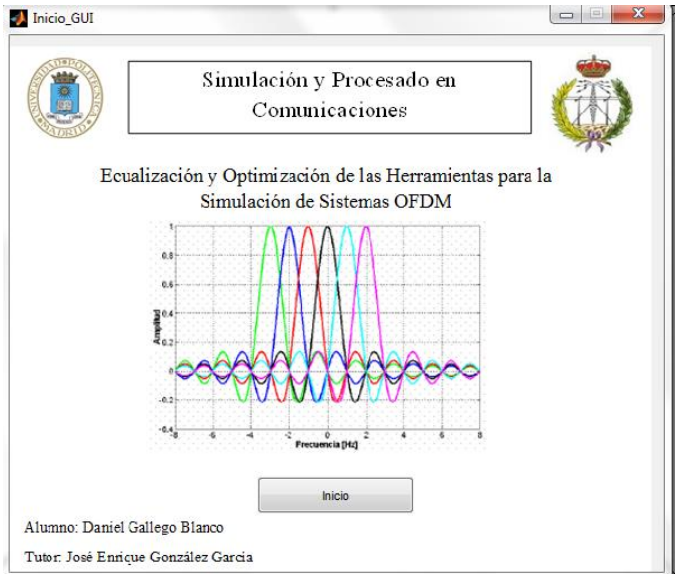

Figura 4.7. Interfaz inicial del programa

Si pulsamos el botón Inicio se nos abrirá otra interfaz donde ya podremos realizar las simulaciones.

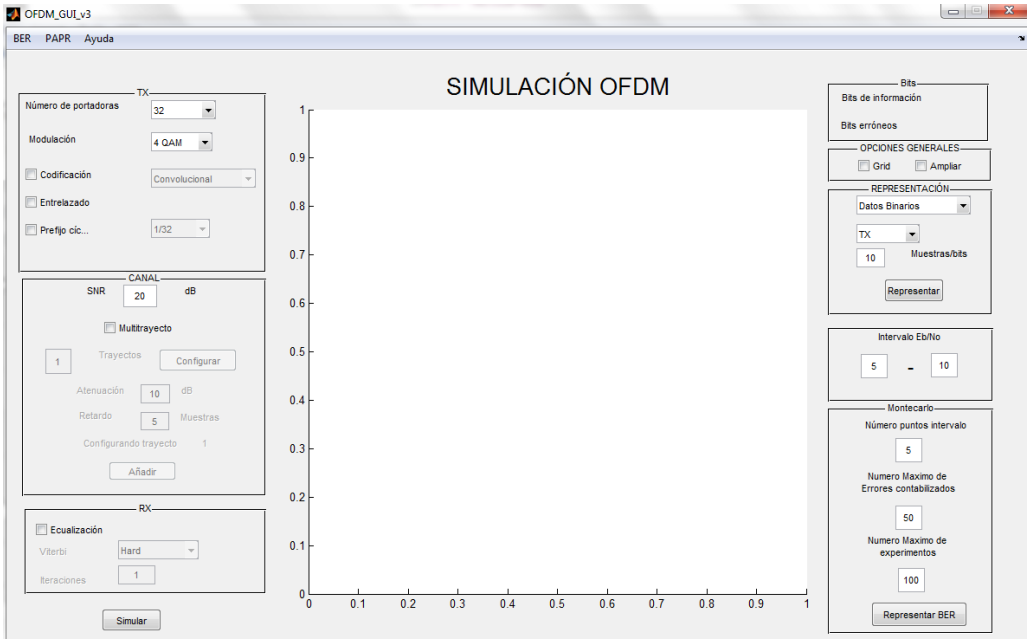

Figura 4.8. Interfaz principal del programa

La parte de la izquierda tenemos la parte de configuración del sistema OFDM, es decir todos los valores necesarios que tenemos que introducir para poder realizar la simulación. Todo lo demás, exceptuando la ayuda en el menú se corresponde con el desarrollo gráfico, donde se representan los resultados de la simulación, se calculan las curvas de la BER, las opciones de representación, etc.

4.2.1. Configuración del esquema OFDM

En este punto se procede a explicar la configuración del esquema OFDM que será utilizado para la simulación. Se divide en tres partes: transmisor, canal y receptor:

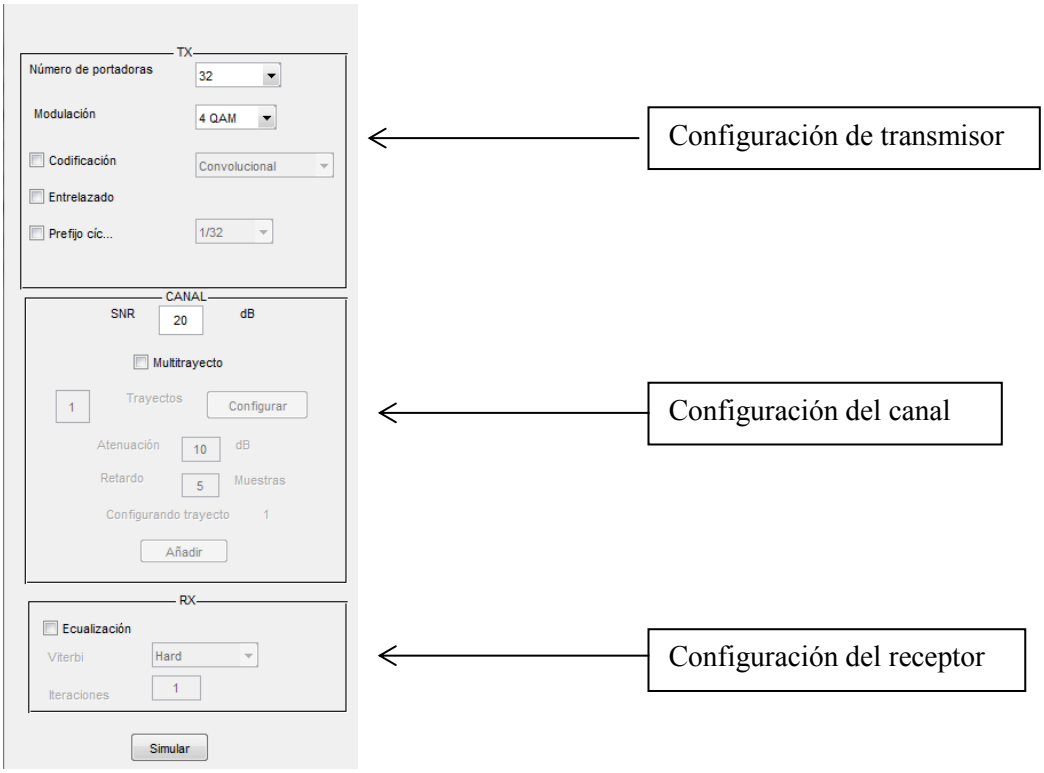

Figura 4.9. Configuración esquema OFDM

4.2.1.1 Configuración del transmisor

En el transmisor se pueden configurar cinco parámetros:

1. Número de portadoras: es el número de portadoras que se utilizarán para transmitir los datos. Se pueden elegir 32, 64, 128, 256, 512, 1024 o 2048. El valor por defecto es 32.

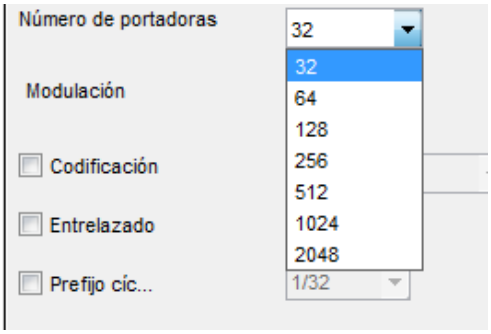

Figura 4.10. Opciones del número de portadoras

2. Modulación: indica el tipo de modulación QAM que se empleará en la simulación. Las opciones son 4, 8, 16, 32, 64 y 128. El valor por defecto es 4.

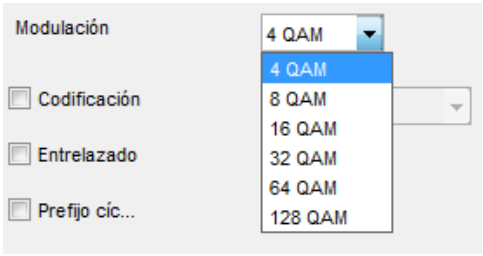

Figura 4.11. Opciones del tipo de modulación.

3. Codificación. Indica si se desea utilizar codificación o no. Por defecto esta opción esta deshabilitada.

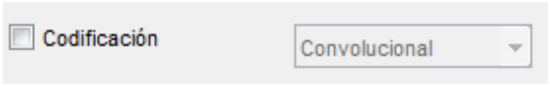

Figura 4.12. Codificación deshabilitada

Se observa que cuando está deshabilitada la codificación no se permite elegir tampoco el tipo de codificación. Si la habilitamos el programa ya nos permite elegir la codificación deseada que puede ser convolucional o turbo. El valor por defecto es convolucional.

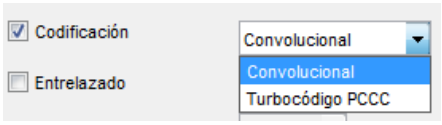

Figura 4.13. Codificación habilitada y tipo de codificación

4. Entrelazado. Permite la opción de entrelazar los bits una vez codificados. Por defecto esta opción está deshabilitada.

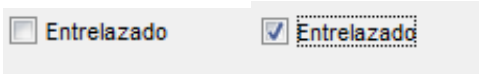

Figura 4.14 Entrelazado

5. Prefijo cíclico. Permite el uso de prefijo cíclico a insertar en el símbolo OFDM. Por defecto deshabilitado. Si se habilita permite elegir el tipo de prefijo (si no está habilitado no se puede elegir el tipo de prefijo) entre los siguientes valores: 1/32, 1/16, 1/8 y 1/4. Por defecto el tipo de prefijo es 1/32.

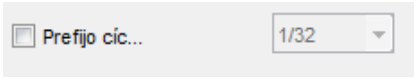

Figura 4.15. Prefijo cíclico deshabilitado

Si se habilita permite elegir el tipo de prefijo (si no está habilitado no se puede elegir el tipo de prefijo) entre los siguientes valores: 1/32, 1/16, 1/8 y 1/4. Por defecto el tipo de prefijo es 1/32.

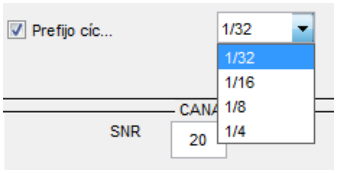

Figura 4.11. Prefijo cíclico habilitado y tipo de prefijo

4.2.1.2 Configuración del canal

En el canal se pueden configurar dos parámetros:

1. SNR: relación señal a ruido. El valor por defecto es 20.

Figura 4.12. Elección SNR

2. Multitrayecto: Permite la elección del multitrayecto. Por defecto está deshabilitado.

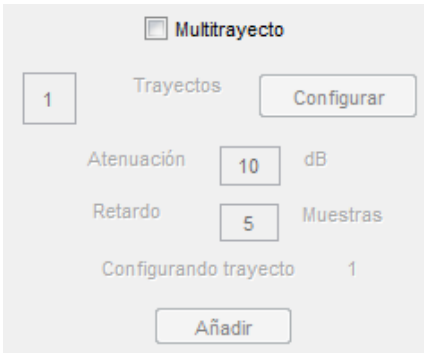

Figura 4.13. Multitrayecto deshabilitado

En la figura se observa que al estar deshabilitado el multitrayecto no se permite configurar ninguna opción del mismo. Si habilitados el multitrayecto estas opciones ya se puede seleccionar el número de trayectos (sin incluir el original, es decir es número de versiones).

Figura 4.14. Multitrayecto habilitado

Y ahora al seleccionar el número de réplicas ya si se habilita las opciones para configurarlas. Por defecto el valor de atenuación es de 10dB y el de retardo 5 muestras.

Figura 4.15. Configuración réplica 1

Una vez configurado el multitrayecto (en este ejemplo se han seleccionado dos) pulsamos el botón añadir para configurar el segundo.

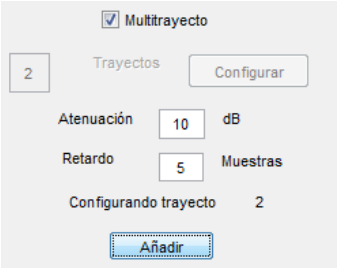

Figura 4.16. Configuración réplica 2

Volveríamos a pulsar el botón de añadir y ya tendríamos configurada nuestro multitrayecto total formado por el original más dos réplicas.

4.2.1.3 Configuración del receptor

En el receptor se encuentro tres parámetros:

1. Ecualizador: indica si se desea utilizar el ecualizador explicado en el tema 3. Por defecto está deshabilitado.

Figura 4.17. Ecualizador habilitado y deshabilitado

2. Decodificador. En el caso de que la codificación haya sido habilitada se permite elegir entre las opciones del decodificador. Si la codificación activada ha sido la convolucional, se permite elegir entre decodificación *hard* o *soft*. Por defecto está seleccionada *hard*.

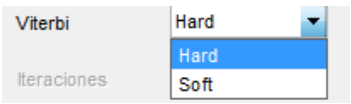

Figura 4.18. Elección tipo decodificador convolucional.

Por el contrario se ha seleccionado turbo codificación se permite seleccionar el número de iteraciones a emplear en la decodificación. El valor por defecto es de 1 iteración.

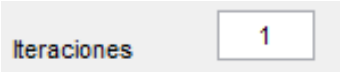

Figura 4.19. Elección número de iteraciones turbo decodificador.

4.2.2. Desarrollo gráfico

Dentro del desarrollo gráfico se encuentran varios bloques que se explican en los siguientes apartados

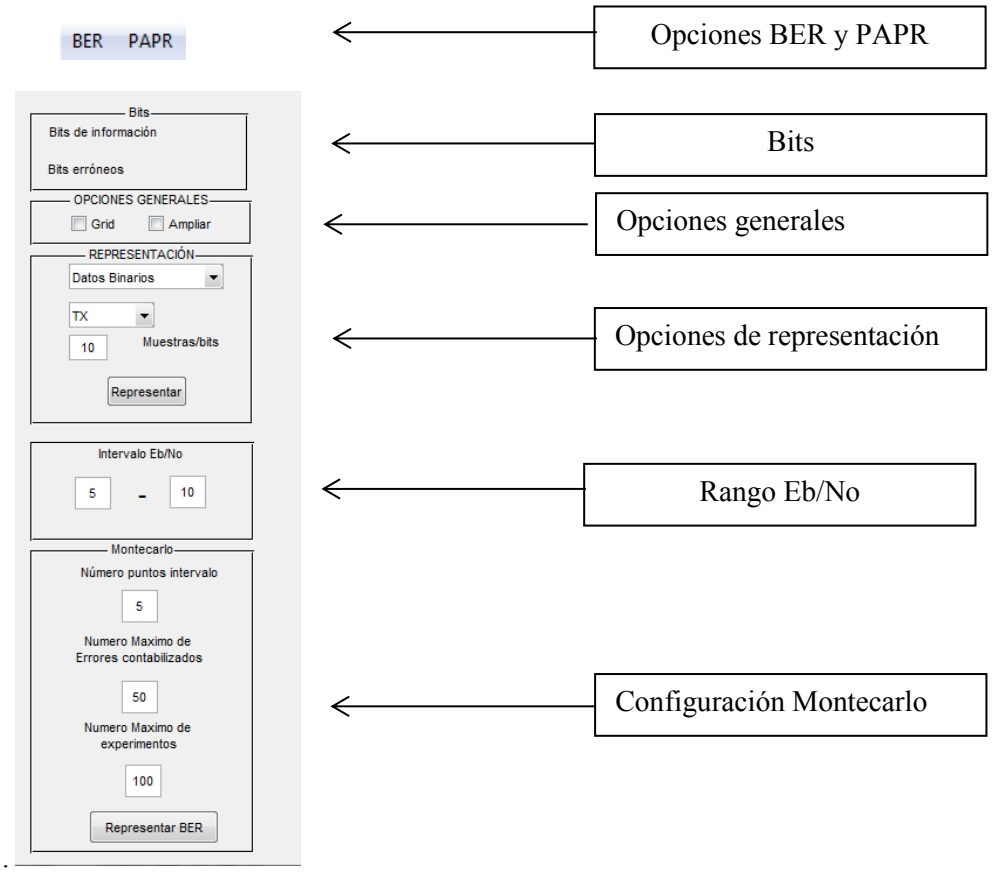

Figura 4.20. Bloque desarrollo gráfico de la interfaz principal.

4.2.2.1. Bits

Esta parte es sólo informativa y por cada simulación nos muestra los bits erróneos y los bits de información enviados. Con bits de información se refiere a los bits útiles, en el caso de no haber codificación todos los bits transmitidos son bits de información, mientras que en el caso de haberla esto no es cierto y los bits de información serán menos que los transmitidos.

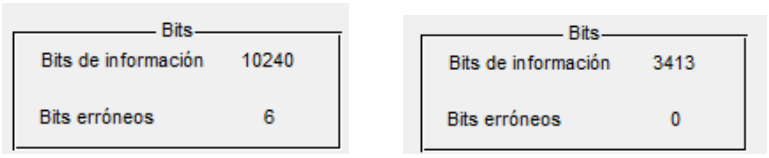

Figura 4.21. Ejemplo bits transmitidos de información y erróneos[15]

4.2.2.2. Opciones generales

Está formado por dos campos:

1. Grid: este campo habilita el grid (la cuadrícula) en todas las representaciones que realicemos. Por defecto está deshabilitado.
2. Ampliar: cuando este campo está habilitado al realizar cualquier representación se nos abre la imagen de Matlab, la cual se puede guardar, editar, etc. Por defecto está deshabilitado.

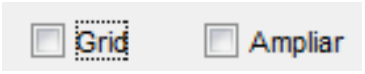

Figura 4.22. Campos de opciones generales

4.2.2.3. Rango Eb/No

Define el rango de Eb/No que se va utilizar tanto el método de Montecarlo, como al representar, por ejemplo, una curva de BER teórica de 4 QAM. Por defecto su valor de 5 a 10.

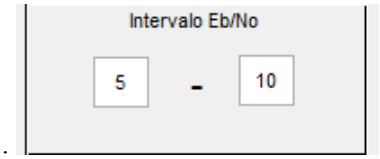

Figura 4.23. Configuración del rango Eb/No

4.2.2.4. Configuración Montecarlo

En este bloque se definen los valores para la realización del método de Montecarlo para calcular la BER bajo unas condiciones de simulación. Para saber en qué rango realizarlo se vale de los valores del bloque anterior (4.2.2.3). Se definen tres parámetros:

1. Número de puntos del intervalo Eb/No. Por ejemplo si el rango de Eb/No va de 5 a 10 y este campo vale 6, los cálculos de realizarán para los valores Eb/No de 5dB, 6dB, 7dB, 8dB, 9dB y 11dB. Por defecto el valor es 5.

[15] Las dos simulaciones se han realizado para 2048 portadoras y una modulación 32 QAM. La primera se ha realizado sin codificación y la segunda con codificación convolucional, habiendo menos bits de información en el segundo caso.

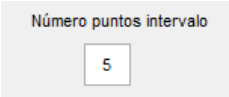

Figura 4.24. Configuración Montecarlo 1

2. Número Máximo de Errores contabilizados: indica el nivel de errores que una vez superado hace que se calcule la BER para el siguiente valor de Eb/No. Valor por defecto 50.

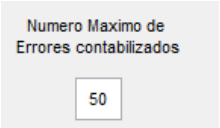

Figura 4.25. Configuración Montecarlo 2

3. Número máximo de experimentos: indica el número máximo de experimentos a realizar para un valor de Eb/No antes de pasar al siguiente. Valor por defecto 100.

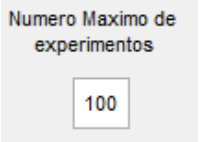

Figura 4.26. Configuración Montecarlo 3

4.2.2.5. Opciones de representación.

En esta parte se definen las formas de representación en los distintos bloques de nuestro esquema OFDM. Hay tres elementos para configurar dicha representación.

1. Tipo de representación: se pueden representar los bits, la constelación IQ, el diagrama de ojos, la señal en el dominio del tiempo y la señal en el dominio de la frecuencia. Por defecto está seleccionada la opción de bits.

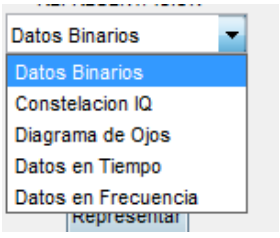

Figura 4.27. Selección modo de representación

2. Modo de representación: Una vez seleccionado el tipo de representación falta seleccionar donde queremos que se representa, si en recepción, en transmisión o en ambos a la vez. Hay una cuarta opción que sólo se puede utilizar cuando se ha seleccionado el modo de representación frecuencia y al realizar la simulación se ha utilizado el ecualizador, si se hace en cualquier otro caso salta un aviso diciendo que no es posible la representación. Este cuarto modo se llama en el programa RX EQ y representa el símbolo en frecuencia después de haber sido ecualizado, se trata de una representación útil para ver el efecto del

ecualizado en el espectro. El valor por defecto para el tipo de representación es transmisión.

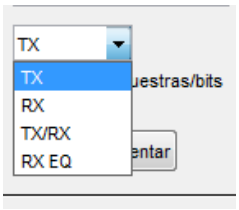

Figura 4.28. Selección tipo de representación

3. Número de bits/muestras: indica el número de bits a mostrar cuando se representan los bits, o el número de muestras de la señal OFDM en el dominio del tiempo. Según si se elige una representación u otra indicará bits o muestras. En las demás representaciones (diagrama de ojos, constelación IQ y dominio de la frecuencia) no influye en absoluto. Su valor por defecto es 10.

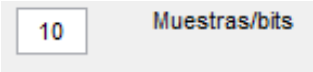

Figura 4.29. Selección número de Muestras/bits

4.2.2.6. BER y PAPR

Primero explicamos las distintas opciones que se pueden realizar dentro del menú BER. Hay cinco opciones:

1. Teórica modulación: representa las curvas teóricas de la BER (en el rango que haya definido en el campo Eb/No) de todas las modulaciones QAM disponibles para simular en el programa. Si se tiene alguna curva almacenada con anterioridad, al representar la curva teórica también se representan las almacenadas.
2. Suavizar curva: sirve para realizar una interpolación entre los puntos que hayan sido calculados de la última BER calculada por el programa.
3. Almacenar Actual: almacena la última curva de BER calculada. Puede almacenarse hasta un máximo de 6.
4. Representar Almacenadas: representa todas las curvas almacenadas en una única gráfica. Útil para realizar comparaciones.
5. Eliminar Almacenadas: elimina todas las curvas almacenadas

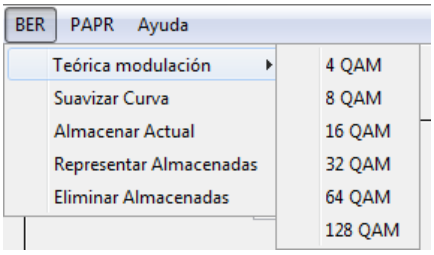

Figura 4.30. Opciones dentro del menú BER

Dentro del menú PAPR hay tres opciones:

1. Representar actual: representa la PAPR de la última simulación obtenida.
2. Ejemplo: representa la PAPR para el caso de un símbolo OFDM con 2, 4, 8, y 16 subportadoras, todas ellas moduladas por una secuencia de símbolos alternados +1 y -1.
3. Teórica CCDF: representa la curva teórica de la probabilidad de superar un cierto umbral de $PAPR_0$ en función del número de subportadoras con el que se esté trabajando; el intervalo de representación para los umbrales de PAPR se selecciona en la opción Intervalo Eb/No.

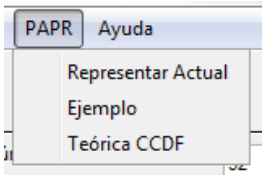

Figura 4.31. Menú PAPR

4.2.3. Menú ayuda

Como se ha dicho anteriormente la interfaz se divide en dos partes quedando la ayuda fuera de ellas. En este menú simplemente se hace un resumen del manual que se acaba de explicar en este capítulo.

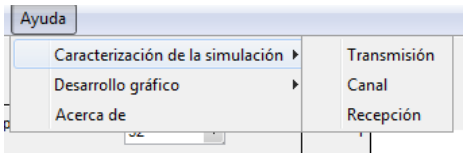

Figura 4.32. Menú ayuda 1

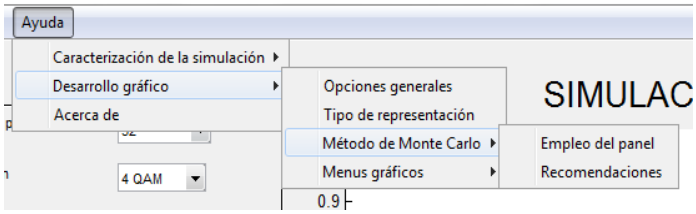

Figura 4.33. Menú ayuda 2

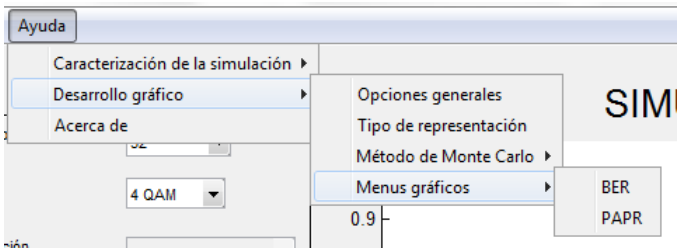

Figura 4.34. Menú ayuda 3

CAPÍTULO 5

RESULTADOS Y ESTUDIOS DE SIMULACIÓN

El capítulo se divide en dos partes. En la primera parte se van a mostrar las diversas formas de representación que nos permite el programa: representación de los bits, en el dominio de la frecuencia, en el dominio del tiempo, etc. En la segunda se van a obtener las curvas de la BER de las distintas codificaciones para poder compararlas entre sí, en concreto se van a realizar tres simulaciones distintas en las que el canal varía para cada una de ellas. Otro objetivo de esta segunda parte es estudiar el funcionamiento de las dos herramientas que tenemos para evitar el multitrayecto: el ecualizador y el prefijo cíclico.

5.1 Simulación Sistema OFDM

En este apartado vamos a prestar atención a las diversas representaciones que realiza el programa en los distintos bloques que conforman nuestro sistema. Todas las simulaciones de los bloques se van a realizar con las mismas condiciones: 2048 portadoras, 16QAM y SNR = 30[16]

5.1.1. Datos binarios

En la figura 5.1 se muestran los 20 primeros bits transmitidos y recibidos.

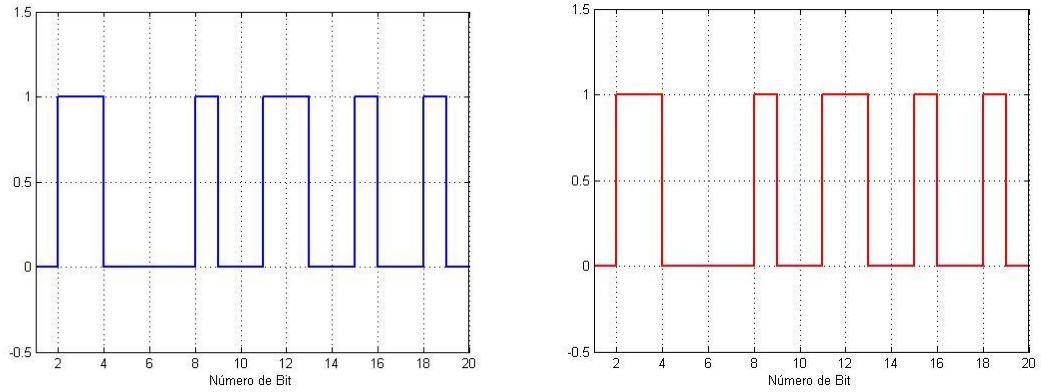

Figura 5.1. Bits transmitidos y recibidos

5.1.2. Datos IQ

Para visualizar los datos una vez modulados en QAM, el programa nos permite dos opciones:

1. Constelación IQ

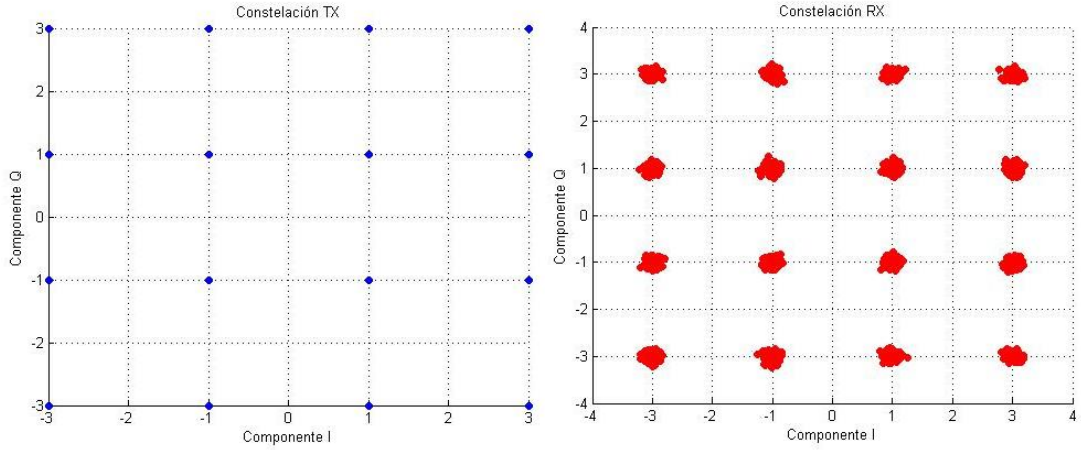

Figura 5.2. Constelación transmitida y recibida

[16] Se ha elegido este nivel de SNR para que se aprecie con nitidez en recepción los niveles del diagrama de ojos. No hay ecualización. ni codificación ni entrelazado.

Al ser la SNR muy buena la constelación se aprecia muy 'limpia', pero puede observarse en la constelación recibida como el ruido produce que los símbolos recibidos no sean exactamente los enviados. Pese a ello los símbolos recibidos son muy similares a los originales, por lo que no se introducen errores.

2. Diagrama de ojos

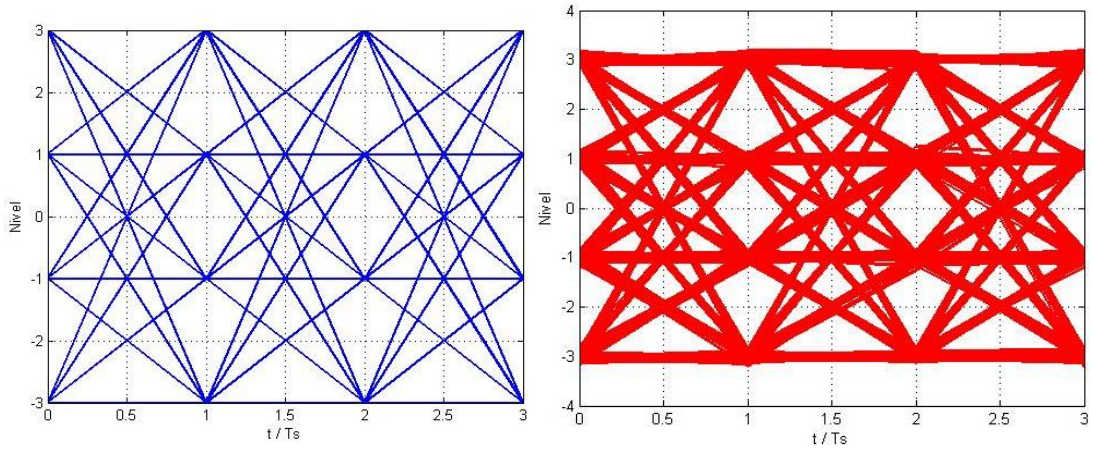

Figura 5.3. Diagrama de ojos en transmisión y recepción

5.1.3. Datos en frecuencia

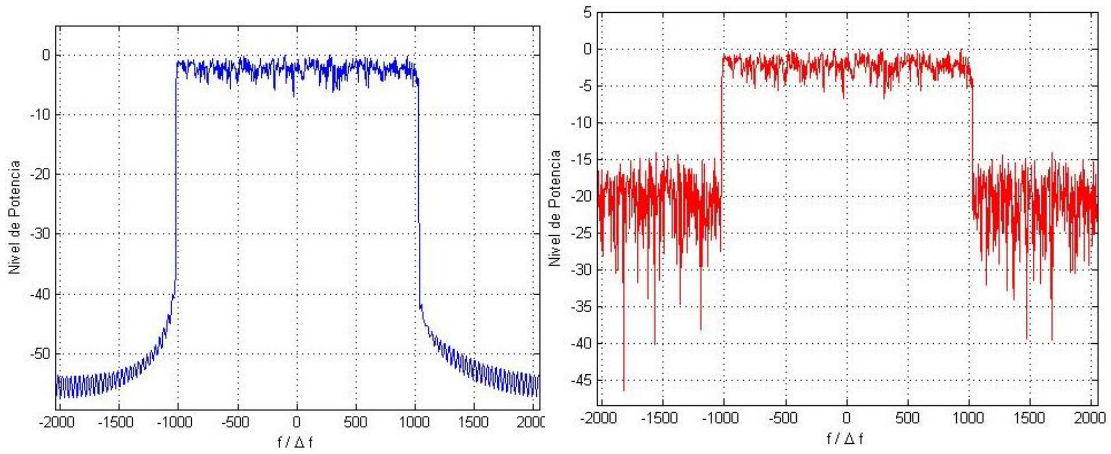

Figura 5.4. Datos en el dominio de la frecuencia

En esta figura se observa la manera en que influye el canal (en este caso solo aporta ruido) en los niveles laterales del espectro. Se aprecia una elevación de ellos.

5.1.4. Datos en el tiempo

En la siguiente figura se representan las 64[17] primeras muestras el símbolo OFDM transmitido:

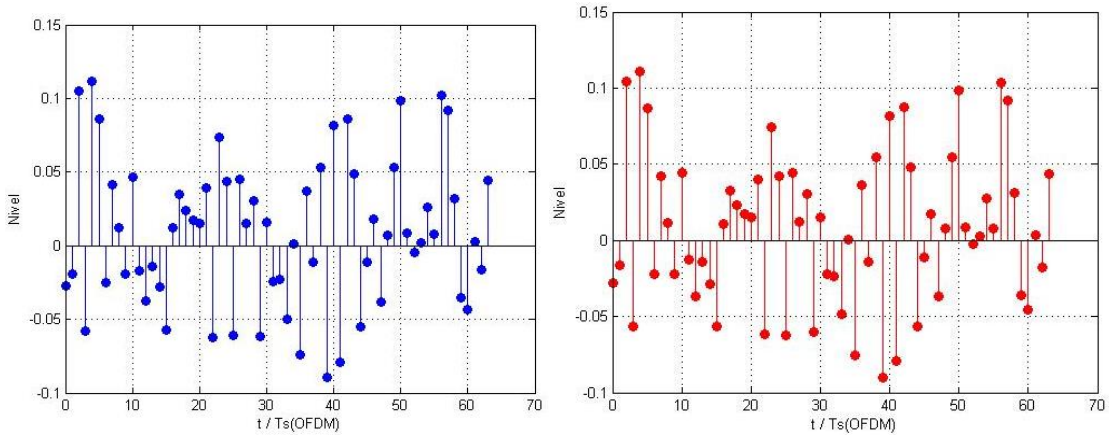

Figura 5.5. Símbolo OFDM en el tiempo

5.2 Simulaciones para el cálculo de la BER

5.2.1 Comparación codificaciones

La primera simulación va a servir para comparar los dos tipos de codificaciones que puede realizar el programa. Las características comunes de las seis simulaciones son: 2048 portadoras, modulación 32 QAM, y que no hay ecualización. El canal únicamente introduce ruido. Para el método de Montecarlo se han seleccionado los siguientes valores:

- Número máximo de errores: 30000.
- Número máximo de experimentos: 100000 para el turbo código con 3 iteraciones.
- Número máximo de experimentos: 10000 para las demás curvas.
- Resolución Eb/No: 1 para el turbo código con 2 y 3 iteraciones[18].
- Resolución Eb/No: 0.5 para las demás curvas.

[17] Por defecto el programa realiza la IFFT con una resolución igual al número de portadoras. El símbolo estaría formado por 2048 muestras, pero únicamente se muestran las 64 primeras para una mejor visualización. Se está mostrando la parte real de cada muestra.

[18] Se ha bajado la resolución con respecto a las otras curvas debido al aumento de tiempo en procesado que requiere la turbo decodificación.

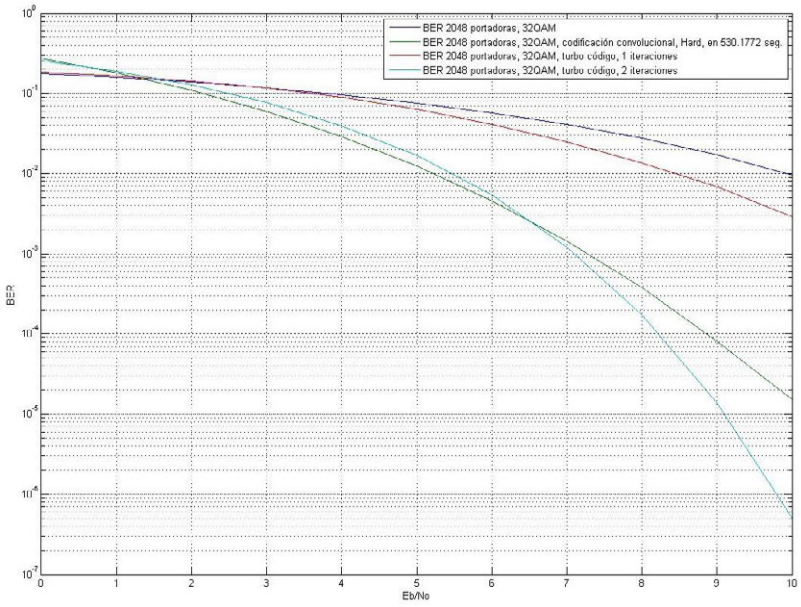

Figura 5.6. Comparación codificaciones 1

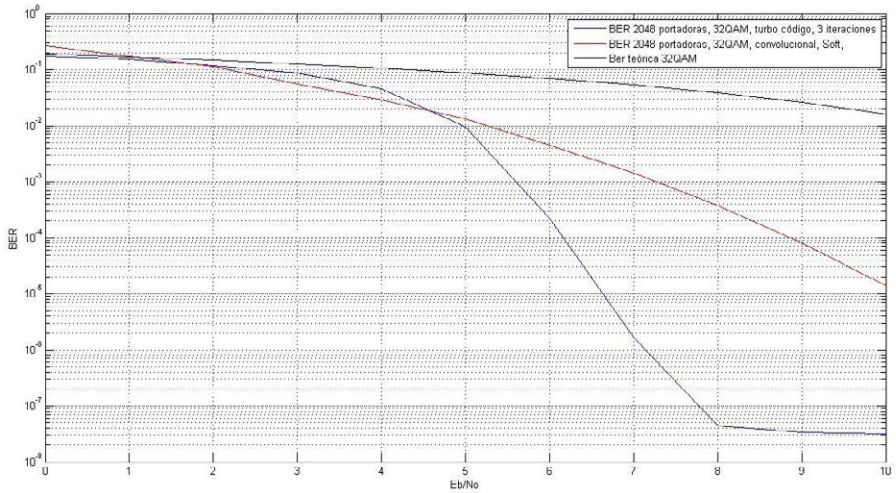

Figura 5.7. Comparación codificaciones 2

En la figura se observa como la curva con mejor comportamiento es la del turbo código con 3 iteraciones, lo cual era de esperar. Si nos fijamos en el punto Eb/N0 = 10 y comparamos todas las demás curvas respecto a la del turbo código con 3 iteraciones se obtienen unas mejoras de:

- 26.31dB respecto al convolucional.
- 11.91dB respecto al turbo código con dos iteraciones.
- 47.93 dB respecto al turbo código con una iteración.
- 54.92 respecto a no utilizar codificación.

Si nos fijamos en la curva teórica de la BER y en la obtenida sin usar codificación observamos que son prácticamente iguales, esto es debido a que en OFDM pese a usar más portadoras se obtienen la misma curva que la teórica. Es una ventaja, ya que podemos usar todas las portadoras que queramos sin empeorar la BER.

5.2.2 Multitrayecto 1

En esta simulación el multitrayecto va a ser 'débil', un multitrayecto en el que no se vean fuertes desvanecimientos, es decir, que presente un espectro más plano y se asemeje más al espectro de un canal únicamente con ruido. Con este ejemplo se quiere comprobar cuando resulta útil el ecualizador y cuando no, a vez que se pone a prueba el prefijo cíclico.[19] El multitrayecto estará formado por la señal original y por cuatro réplicas cuyas características son[20]:

- Réplica 1: Atenuación:10dB, Retardo: 5 muestras
- Réplica 2: Atenuación:13dB, Retardo: 10 muestras
- Réplica 3: Atenuación:15dB, Retardo: 13 muestras
- Réplica 4: Atenuación:20dB, Retardo: 20 muestras

En todas las simulaciones se han utilizado 2048 portadoras y una modulación 32 QAM.

En la figura 5.6 se representa la BER para las distintas codificaciones (y sin codificar) para las características de canal descritas por Multitrayecto 1 sin el uso de prefijo cíclico ni ecualizador. Los parámetros para el método de Montecarlo son:

- Número máximo de errores: 90000
- Número máximo de experimentos: 6000
- Resolución Eb/No: 0.5

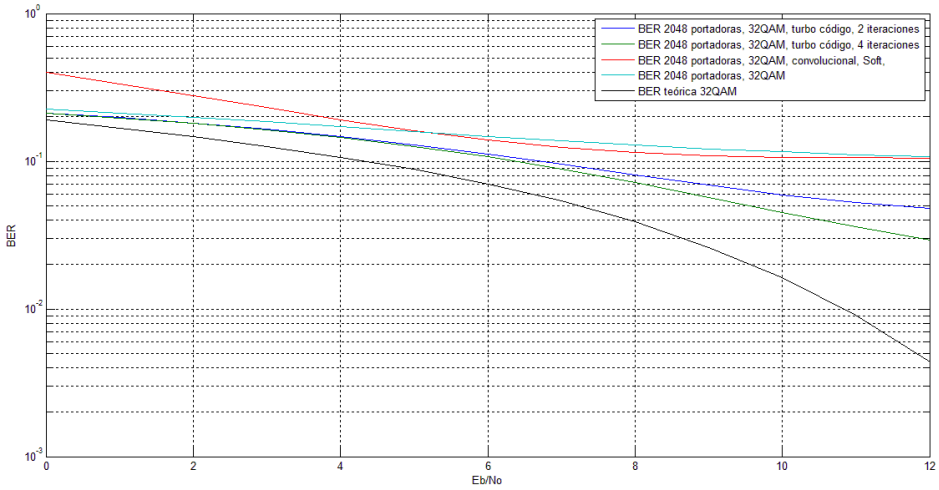

Figura 5.8. BER Multitrayecto 1

En la gráfica se observa que el comportamiento de la curva del código convolucional es muy semejante al de la curva sin utilizar codificación. El turbo código es el que mejor se comporta, obteniendo una mejora (Eb/No=12dB) para dos iteraciones de 3dB y de 5.22dB para cuatro iteraciones con respecto a la curva del convolucional.

A continuación se prueba el funcionamiento del prefijo cíclico únicamente. Los parámetros para el método de Montecarlo son:

[19] El prefijo cíclico utilizado será 1/32, el cual nos da 2048/32=64 muestras repetidas, que es mayor que el número máximo de retardos que es 20.

[20] A partir de ahora al multitrayecto formado por la señal original y estas cuatro réplicas se le nombrará simplemente como Multitrayecto 1.

- Número máximo de errores: 30000
- Número máximo de experimentos: 10000
- Resolución Eb/No: 0.5

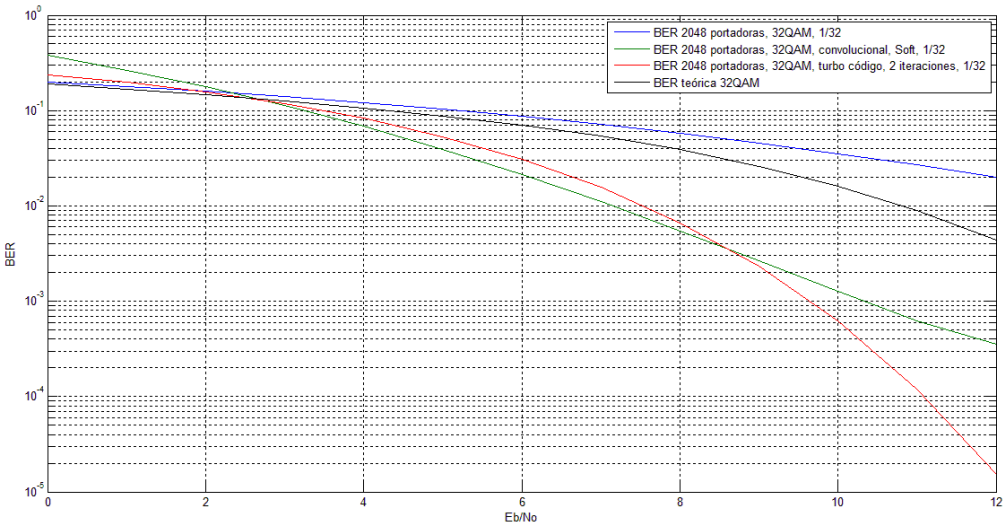

Figura 5.9. BER Multitrayecto 1 con prefijo 1/32

Se observa que el turbo código obtiene el mejor resultado, comportándose de forma muy parecida al convolucional hasta una Eb/No de 8.5dB; a partir de ese valor la curva del turbo código cae más deprisa obteniendo una máxima diferencia para de 13.68dB entre ambas curvas para una Eb/No de 12dB. Si comparamos los resultados de esta gráfica con los de la anterior[21] se obtiene una diferencia máxima entre las curvas de la BER para el turbo código de 35.23dB y de 24.56dB para el convolucional.

Se representa por separado los resultados de la BER para el caso del uso del turbo código con 4 iteraciones. Los valores para la simulación son:

- Número máximo de errores: 15000
- Número máximo de experimentos: 80000
- Resolución Eb/No: 1

[21] Se están comparando las curvas de las mismas codificaciones para el caso de no usar prefijo y de sí usarlo.

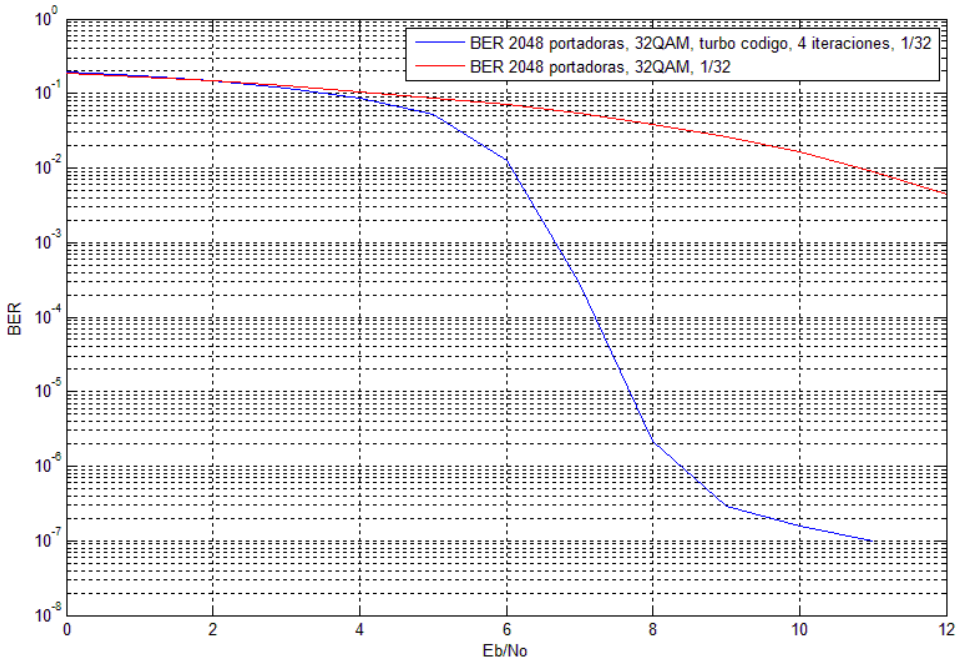

Figura 5.10. BER Multitrayecto 1 con prefijo 1/32, turbo código 4 iteraciones

Lo primero que hay que decir sobre la figura 5.10 es que no se han conseguido valores para una Eb/No[22] de 12dB. Llama la atención como la curva cae muy bruscamente entre los valores de Eb/No de 6 y 8 dB, cayendo casi 4 décadas, para luego estabilizarse y caer de forma menos abrupta. Para una Eb/No de 11dB, el turbo código con 4 iteraciones presenta una mejora de 30dB con respecto a 2 iteraciones y de 37.8 respecto al convolucional. Comparando el turbo código con 4 iteraciones con y sin prefijo se aprecia una mejora de 54.8dB.

En la siguiente gráfica se analiza el empleo del ecualizador para este tipo de desvanecimientos. Los parámetros para el método de Montecarlo son:

- Número máximo de errores: 90000
- Número máximo de experimentos: 6000
- Resolución Eb/No: 0.5

[22] La simulación llevada a cabo ha durado 23 horas y con 80000 simulaciones no se ha conseguido ni un error para poder calcular la BER. El mayor problema de los turbo códigos es el tiempo de simulación que requieren, a mayor número de iteraciones el tiempo de simulación aumenta considerablemente.

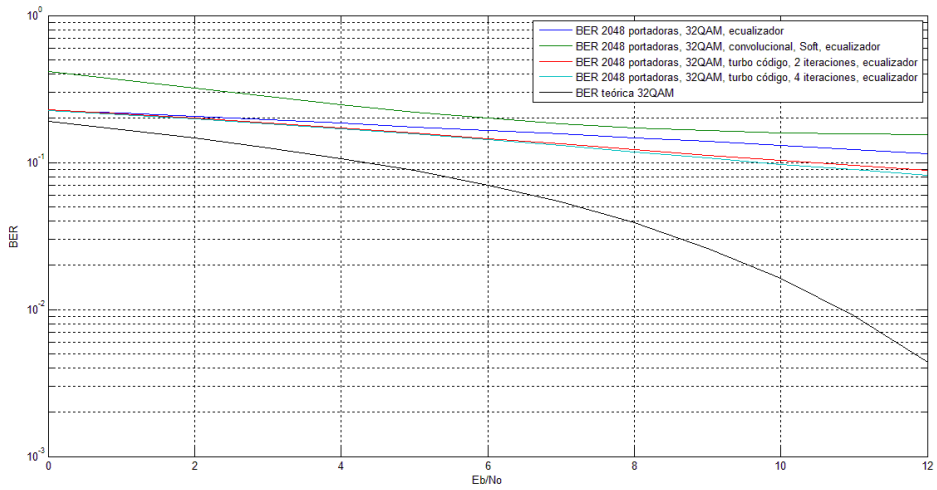

Figura 5.11. BER Multitrayecto 1 con ecualizador

Observando la gráfica se llega a la conclusión de las curvas con peores que en el caso de no utilizar ecualizador. Para este tipo de desvanecimientos el ecualizador no es útil.

Por último falta comprobar el comportamiento del ecualizador y el prefijo juntos. Los parámetros para el método de Montecarlo son:

- Número máximo de errores: 90000
- Número máximo de experimentos: 6000
- Resolución Eb/No: 0.5

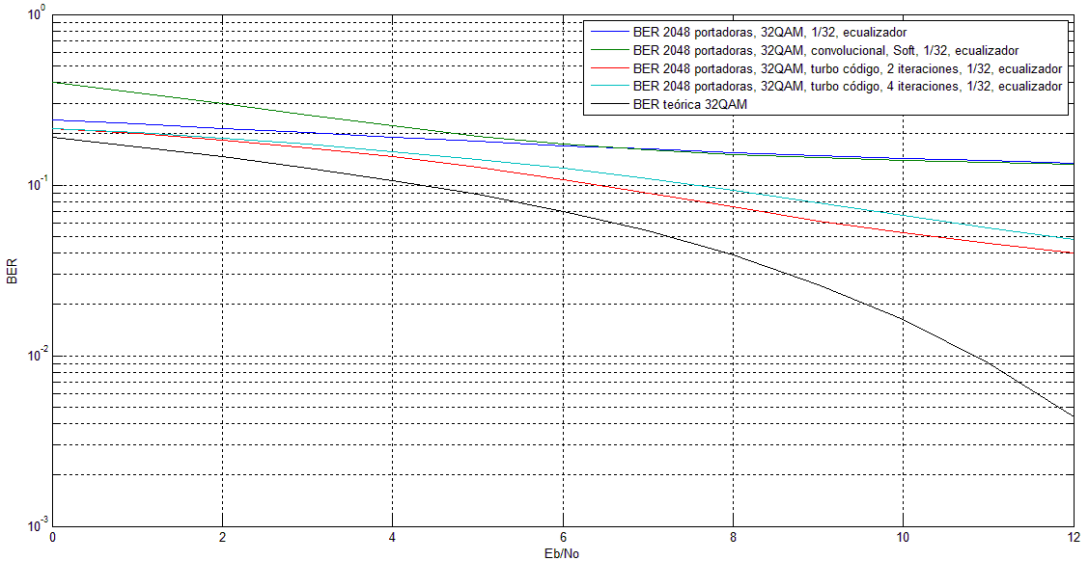

Figura 5.12. BER Multitrayecto 1 con ecualizador y prefijo cíclico

Pese a que se han obtenido mejores resultados que en la simulación con el ecualizador únicamente. Los resultados son bastante peores que con los conseguidos en la simulación con el prefijo cíclico en solitario. Por lo tanto ahora sí se puede afirmar que el ecualizador no es práctico para este tipo de desvanecimientos, mientras que el prefijo cíclico resulta una herramienta muy útil para combatirlo.

En la última figura de esta simulación se presenta el espectro de la señal recibida.

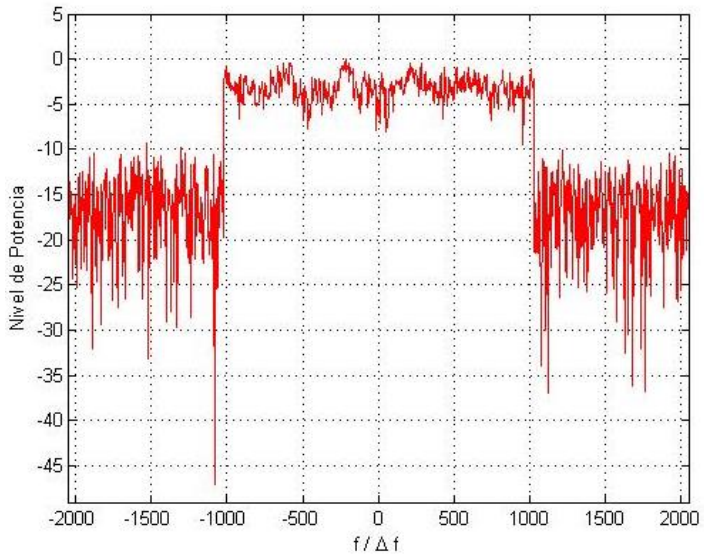

Figura 5.13. Espectro recibido Multitrayecto 1[23]

Se observa que los desvanecimientos (picos) no son muy grandes, asemejándose más a ruido. Era el espectro buscado para probar la utilidad del ecualizador para cierto tipo de desvanecimientos.

5.2.3 Multitrayecto 2

En este caso el multitrayecto va a ocasionar que aparezcan claramente desvanecimientos en el espectro en recepción. En este apartado se va a poner a prueba los dos mecanismos que tenemos para contrarrestar el multitrayecto: el ecualizador y el prefijo cíclico[24]; se van a probar por separado y juntos para comprobar la manera en la cual se obtiene mejor resultado. El multitrayecto estará formado por la señal original y por tres réplicas cuyas características son[25]:

- Réplica 1: Atenuación:3dB, Retardo: 2 muestras
- Réplica 2: Atenuación:6dB, Retardo: 3 muestras
- Réplica 3: Atenuación:10dB, Retardo: 10 muestras

En todas las simulaciones se están utilizando 2048 portadoras y una modulación 16 QAM.

En la siguiente figura se muestran los resultados de la BER obtenidos, no se ha utilizado ni ecualización ni prefijo cíclico. Los parámetros para el método de Montecarlo son:

- Número máximo de errores: 90000
- Número máximo de experimentos: 6000
- Resolución Eb/No: 0.5

[23] Este espectro se ha realizado con una SNR=20.
[24] El prefijo cíclico utilizado será 1/32, el cual nos da 2048/32=64 muestras repetidas, que es mayor que el número máximo de retardos que es 10.
[25] A partir de ahora al multitrayecto formado por la señal original y estas tres réplicas se le nombrará simplemente como Multitrayecto 2.

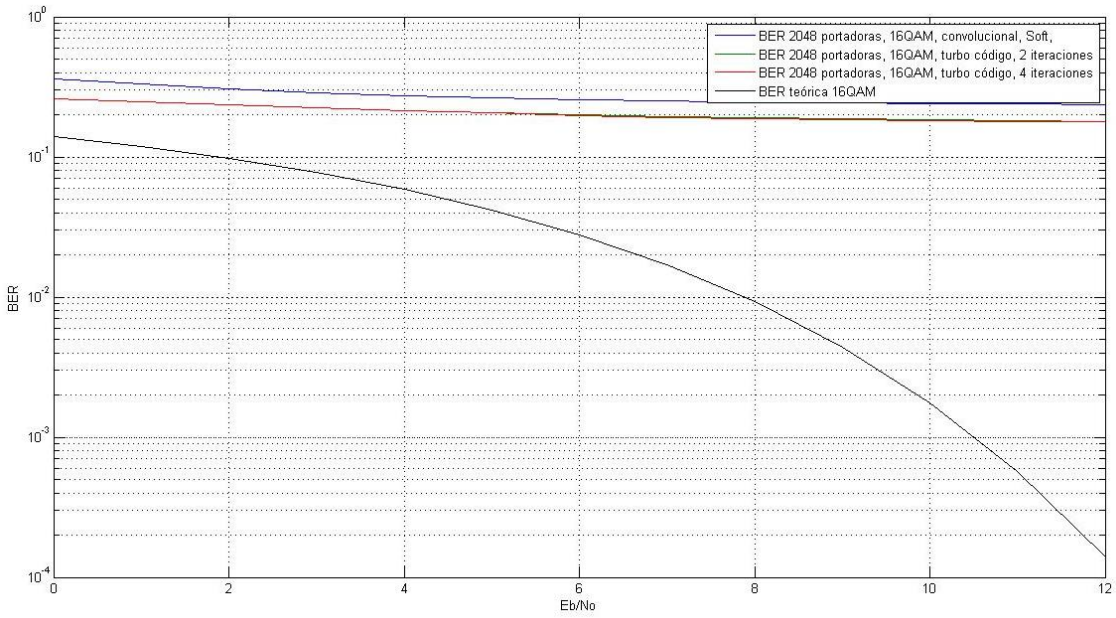

Figura 5.14. BER Multitrayecto 2

En las dos gráficas anteriores se observa que sin el uso de ningún método para combatir el multitrayecto la BER de las distintas codificaciones es muy plana, habiendo apenas diferencia entre las distintas codificaciones. También se aprecia que la que mejor resultados obtiene es la turbo codificación sin haber apenas diferencia entre utilizar 2 o 4 iteraciones.

A continuación se pone a prueba el prefijo cíclico. Los parámetros para el método de Montecarlo son:

- Número máximo de errores: 90000
- Número máximo de experimentos: 6000
- Resolución Eb/No: 0.5

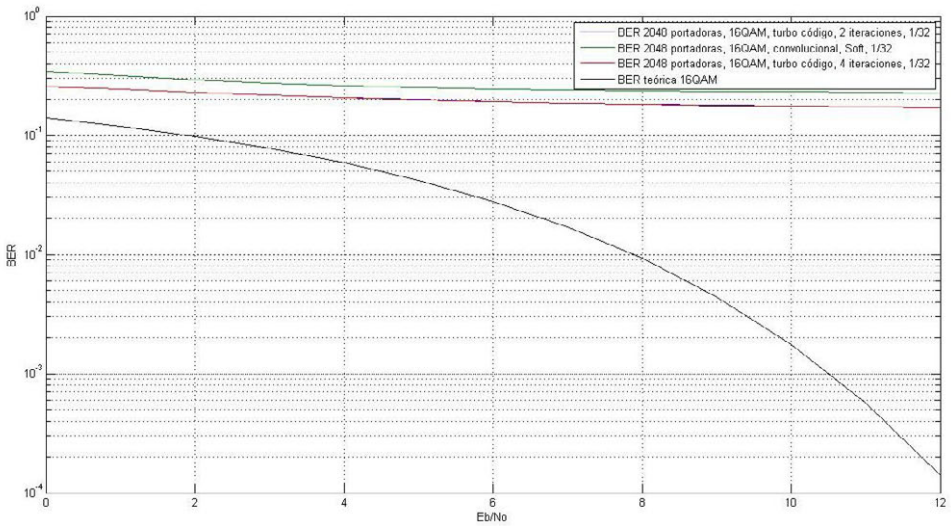

Figura 5.15. BER Multitrayecto 2 con prefijo

Los resultados son prácticamente iguales que los obtenidos sin ecualización, por lo que para esta simulación se puede afirmar que el prefijo cíclico por sí solo no puede corregir un multitrayecto que provoca un desvanecimiento tan fuerte en el espectro.

Ahora pasamos a probar el ecualizador. Los parámetros para el método de Montecarlo son:

- Número máximo de errores: 90000
- Número máximo de experimentos: 6000
- Resolución Eb/No: 0.5

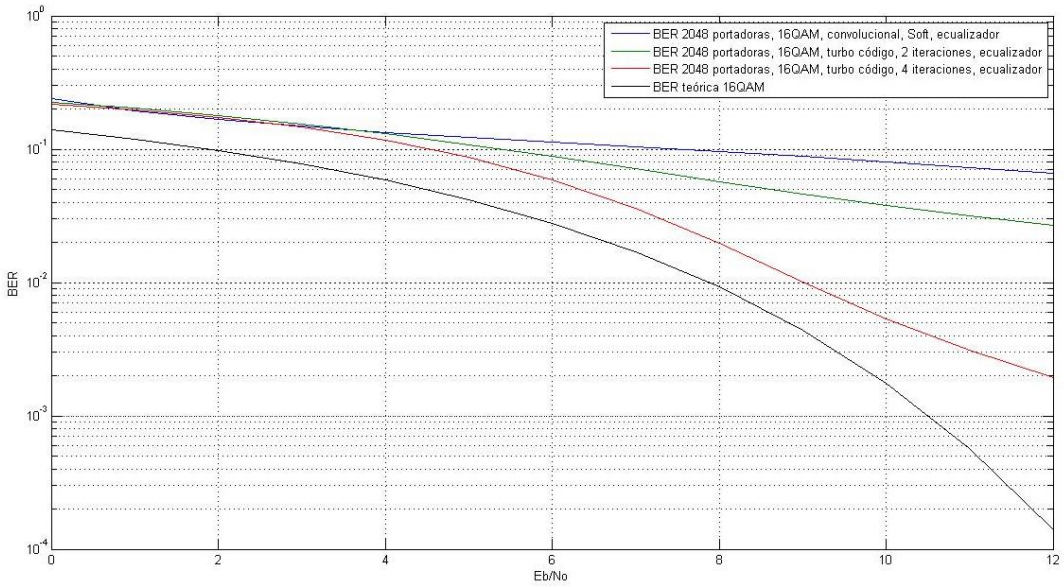

Figura 5.16. BER Multitrayecto 2 con ecualizador

Con ecualizador ya se empieza a observar una mejor en la BER para todas las curvas. La mayor mejora se encuentra con el turbo código y 4 iteraciones, si nos fijamos para una Eb/No de 12dB hay una mejora en la BER de aproximadamente 19.54dB (ha caído dos décadas). En las demás curvas se observa una ganancia aunque significativamente menor, por ejemplo para la codificación convolucional la caída es de 5.85dB.

Por último probamos conjuntamente el uso del prefijo y del ecualizador. Los parámetros para el método de Montecarlo son:

- Número máximo de errores: 60000
- Número máximo de experimentos: 10000
- Resolución Eb/No: 0.5

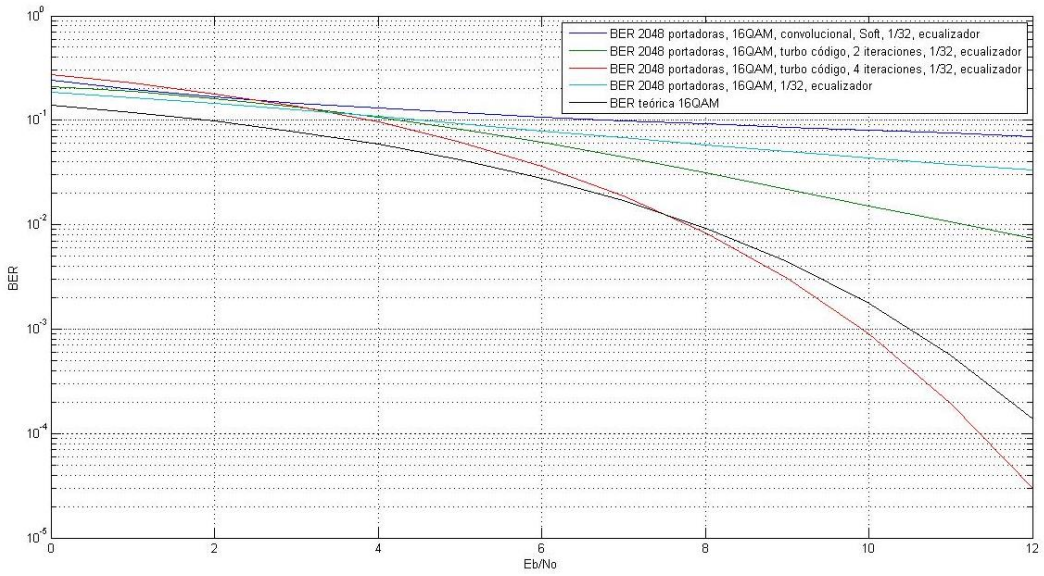

Figura 5.17. BER Multitrayecto 2 con prefijo 1/32 y ecualizador

En el uso conjunto de ecualizador y prefijo cíclico se obtienen los mejores resultados para la BER, de hecho es la primera vez que se supera a la curva teórica, consiguiéndolo el turbo código con 4 iteraciones a partir de una Eb/No de aproximadamente 7.5dB. Si nos fijamos en el caso anterior en el mejor de los casos, observamos que para una Eb/No de 12dB el turbo código con 4 iteraciones presenta una mejora de 18.23dB respecto al uso de sólo el ecualizador, y de 37.78dB respecto a no utilizar ecualizador. Si nos fijamos en la curva del convolucional prácticamente no hay mejora con respecto al caso anterior. El turbo código con dos iteraciones presenta una mejora de 6dB respecto a la gráfica 5.16 y de 14.1dB respecto a la 5.15.

A continuación se muestra el espectro de la señal recibida para comprobar que con el Multitrayecto 2 se obtienen desvanecimientos considerables.

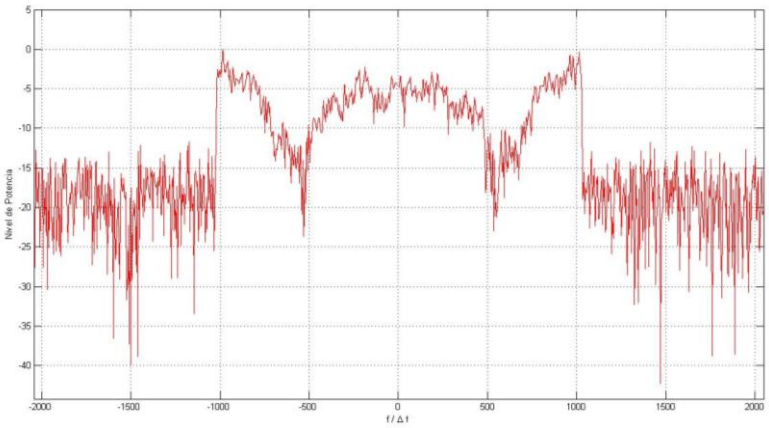

Figura 5.18. Espectro recibido Multitrayecto 2[26]

[26] Este espectro se ha realizado con una SNR=20.

Se comprueba en la figura 5.18 que se producen grandes desvanecimientos. Igual que en la figura 5.4 se observaba únicamente la influencia del ruido, en esta se aprecia claramente los efectos que tiene nuestro esquema de canal: el ruido en los laterales del espectro, y el multitrayecto que provoca los desvanecimientos.

Por último en la siguiente imagen se va a mostrar el espectro de la señal recibida después de pasar por ecualizador.

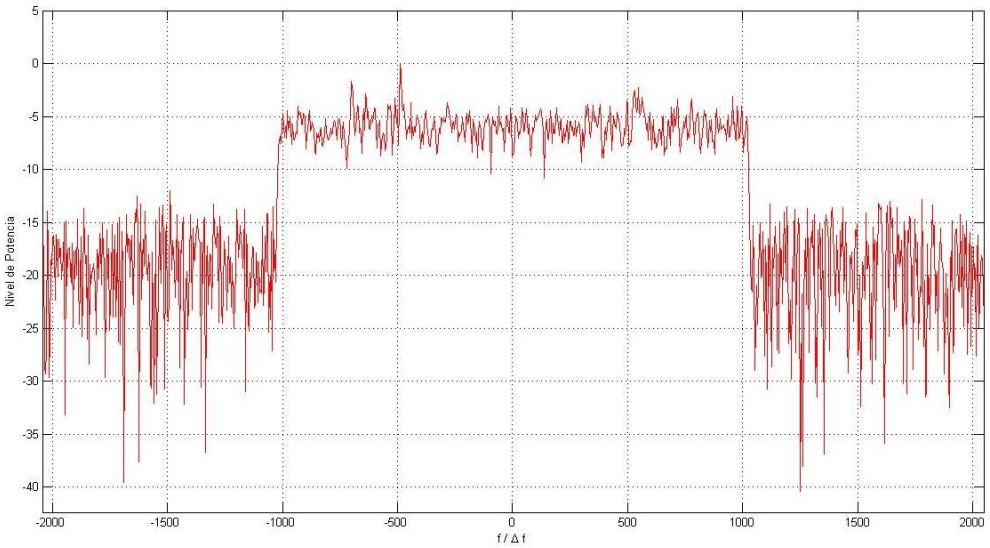

Figura 5.20. Espectro ecualizado recibido Multitrayecto 2[27]

El ecualizador ha suavizado el espectro corrigiendo los desvanecimientos producidos por el multitrayecto.

[27] Este espectro se ha realizado con una SNR=20.

CAPÍTULO 6

CONCLUSIONES Y POSIBLES AMPLIACIONES

Una vez realizadas las simulaciones en el capítulo anterior, ya estamos en condiciones de extraer las conclusiones obtenidas mediante el desarrollo de este proyecto. También se va a proceder a dar unas posibles líneas de ampliación para versiones futura del programa desarrollado.

6.1 Conclusiones

El desarrollo de este proyecto se ha centrado en ampliar el proyecto de [Vilca 11] mediante el uso de nuevas formas de codificación (los turbo códigos) y de la ecualización. Todo ello para poder bajar las tasas de bits erróneas en canales más complicados. Las conclusiones obtenidas respecto a la mejora en las curvas de la BER son:

- La turbo codificación ha resultado una herramienta muy potente para bajar las tasas de bits erróneos, ya que a base de subir las iteraciones en el decodificador se consiguen grandes ganancias en las curvas de la BER.

- El ecualizador se ha mostrado muy efectivo para combatir el multitrayecto cuando éste produce grandes desvanecimientos en el espectro. En concreto, combinado con el prefijo cíclico es cuando se han obtenido los mejores resultados.

- No ha sido práctico el uso del ecualizador para combatir el multitrayecto cuando no produce grandes desvanecimientos, asemejándose el espectro recibido al que tendríamos si sólo hubiera ruido. En este caso la herramienta más útil ha sido la utilización del prefijo cíclico en solitario.

Respecto a las conclusiones obtenidas respecto al programa utilizado para la simulación del esquema OFDM, las conclusiones más importantes son:

- Matlab ha resultado una herramienta muy potente , útil y apropiado para la simulación de nuestro esquema

- La turbo decodificación ha resultado muy lenta en comparación con la decodificación convolucional, sobre todo a partir de tres o más iteraciones. Debido a ello en las gráficas se representaba como máximo hasta un valor Eb/No de 12 dB y poder comparar los resultados obtenidos con las distintas codificaciones.

- Para poder obtener representaciones para mayores valores de Eb/No y estudiar mejor el comportamiento de los turbo códigos y sus iteraciones sería necesario el uso de un ordenador más potente con mayor capacidad de procesado.

6.2 Posibles ampliaciones

Respecto a las posibles ampliaciones que se puedan realizar a este proyecto.

- Implementación de otros esquemas de turbo codificación. En este proyecto ha sido implementado el esquema PCCC, dejando sin utilizar los esquemas SCCC y HCCC.

- Implementación de un ecualizador basado en algoritmo ZF (forzado a cero)

- Uso de modulación adaptiva. Útil para cuando se tiene una alta tasa de bits erróneos. Por ejemplo si se está utilizando una modulación 64 QAM para una SNR concreta y se detecta

una alta tasa de bits erróneos, utilizar una modulación más sencilla como pueda ser una 16 QAM para las subportadoras más afectadas por el canal.

- Implementación de técnicas MIMO (*Multiple Input Multiple Output*). MIMO aprovecha fenómenos físicos como la propagación multicamino para incrementar la tasa de transmisión y reducir la tasa de error. Aumenta la eficiencia espectral de un sistema de comunicación inalámbrica por medio de la utilización del dominio espacial.

- Implementación de técnicas para la reducción de la PAPR. Uno de los problemas de la modulación OFDM es su elevada PAPR que causa problemas de distorsión no lineal.

- Implementación SC-FDMA (Simple Carrier – Frequency Division Multiple Access). Se trata de una variante del esquema OFDMA, pero efectuando una precodificación de los símbolos a transmitir previa al proceso de transmisión OFDM. Con ellos se consigue la reducción de la PAPR, la mejora de la ecualización en frecuencia y la capacidad de proporcionar una asignación de banda flexible.

- Optimización de las funciones implementadas para poder llevar a cabo las simulaciones a una mayor velocidad, sobre todo para poder calcular las curvas de la BER en menos tiempo.

PRESUPUESTO

Para poder calcular el presupuesto de este proyecto partimos del caso de un Ingeniero Técnico de Telecomunicaciones que recibe un sueldo de 40 euros/hora, la búsqueda de información, el desarrollo del programa y en la obtención de las simulaciones, y de 20 euros/hora en la redacción de la memoria. En la tabla 5.1 muestra las horas dedicadas y su precio:

TAREA	HORAS	Precio de la hora	Precio total
Búsqueda de información	170	40	6800
Desarrollo del programa	200	40	8000
Simulaciones y ensayos	110	40	4400
Redacción	80	20	1600
Total	560		20800

Tabla 6.1. Presupuesto por horas de trabajo

Por lo que por horas dedicadas obtenemos un precio de 20800 euros. A este precio habrá que añadir otros conceptos que se incluyen en la siguiente tabla:

Concepto	Precio
Papelería	20
Reprografía	200
Otros (PC, licencia Matlab, luz, etc,)	4000
Total	4220

Tabla 6.2. Presupuesto por otros conceptos

Por lo tanto el presupuesto total es de 25.020 euros, que una vez aplicado el impuesto del IVA del 21% se llega a un presupuesto final de 30.274,2 euros.

BIBLIOGRAFÍA

[Agusti 10] R. Agusti Comes, F. Bernardo Álvarez, F. Casadevall Palacio, R. Ferrús ferre, J. Pérez Romero, O. Sallent Roig, "LTE Nuevas Tendencias en Comunicaciones Móviles", Fundación Vodafone España, 2010.

[Artés 07] A. Artés, F. Pérez González, R. López, C. Mosquera, F. Perez Cruz, "Comunicaciones digitales", Pearson Prentice Hall, 2007.

[Barbulescu 04] Sorin Adrian Barbulescu, "What a wonderfull turbo world" ISBN: 09580520 0 X. Versión 1.2, 2004.

[Benedetto 87] Sergio Benedetto, Ezio Biglieri, Valentino Castellani, "Digital Transmission Theory", Prentice-Hall International, 1987.

[Berrou 93] C. Berrou, A. Glavieux, P. Thitimajshima, "Near Shannon limit error-correcting coding and decoding: Turbo-codes. 1", IEEE, 1993.

[CDMA2000 2009] 3GPP2 C.S0002-E v1.0, 2009.

[Glover 04] I. Glover, P. Grant, "Digital Communications", Second Edition, Pearson Prentice Hall, 2004.

[Hanzo 2000] L. Hanzo, J. Woodward, "Comparative Study of Turbo Decoding Techniques: An Overview", IEEE Transactions on Vehicular Tecnhnology. Vol. 49,pp. 2208-2233, 2000.

[Ingvarsson 98] P. Ingvarsson, H.Svenell, "Error perfomance of turbo codes", 1998.

[Jeruchim 84] Michael C. Jeruchim, "Techniques for Estimating the Bit Error Rate in the Simulation of Digital Communication Systems", IEEE 1984.

[Jeruchim 00] Michael C. Jeruchim, Philip Balaban, K. Sam Shanmugan, "Simulation of Communication System", KA/PP, 2000.

[LTE 2011] ETSI TS 136 212 V10.1.0 (2011-04), 3GPP TS 36.212 version 10.1.0 Release 10, 2011.

[Mackay 06] David J.C. Mackay, "Information Theory, Inference and Learning Algorithms", Cambridge, 2006.

[Prasad 04] R.Prasad, "OFDM for Wireless Communications Systems", Artech House, 2004.

[Pietrobon 98] S. Pietrobon, S. Barbulescu, "TURBO CODES: a tutorial on a new class of poweful error correcting coding schemes" ITR University Australia, 1998.

[Proakis 01] Jhon G. Proakis, Digital Communications, Fourth Edition, McGraw Hill, 2001.

[Rábanos 08] José María Hernando Rábanos, "Transmisión por radio", Centro de estudios Ramón Areces, 2008.

[Sesia 09] S. Sesia, I. Toufik, M. Baker, "LTE – The UMTS Long Term Evolution: From Theory to Practice", John Wiley & Sons, 2009.

[Vilca 11] Diego Fermín Vilca Ureta, "Ampliación y Optimización de las Herramientas para la Simulación de Sistemas OFDM", PFC, Escuela Universitaria de Ingeniería Técnica de Telecomunicaciones, Universidad Politécnica de Madrid, 2011.

[Viterbi 79] A.J. Viterbi, J.K Omura, "Principles of Digital Communication and Coding", McGraw-Hill, 1979.